ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ
ΕΡΓΑΣΤΗΡΙΟ ΥΠΟΛΟΓΙΣΤΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

Σύστημα οπτικοακουστικής επικοινωνίας και καταγραφής με το πρωτόκολλο WebRTC

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Σοφοκλής Β. Καραβέλλας

Επιβλέπων : Παναγιώτης Δ. Τσανάκας
Καθηγητής Ε.Μ.Π.

Αθήνα, Ιούνιος 2018

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ
ΕΡΓΑΣΤΗΡΙΟ ΥΠΟΛΟΓΙΣΤΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

Σύστημα οπτικοακουστικής επικοινωνίας και καταγραφής με το πρωτόκολλο WebRTC

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Σοφοκλής Β. Καραβέλλας

Επιβλέπων : Παναγιώτης Δ. Τσανάκας
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 18η Ιουνίου 2018.

.....	.....	.....
Παναγιώτης Τσανάκας	Κιαμάλ Πεκμεστζή	Ηλίας Μαγκλογιάννης
Καθηγητής Ε.Μ.Π	Καθηγητής Ε.Μ.Π	Αν. Καθηγητής ΠΑ.ΠΕΙ.

Αθήνα, Ιούνιος 2018

.....

Σοφοκλής Β. Καραβέλλας

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Περίληψη

Με την εξέλιξη του διαδικτύου και την σταδιακή επέκταση του στην καθημερινή ζωή των ανθρώπων παρατηρείται πως χρησιμοποιείται όλο και περισσότερο για την επικοινωνία μεταξύ τους. Η χρήση του μας δίνει επιπλέον τη δυνατότητα να προσθέσουμε και εικόνα στην επικοινωνία μας. Για την δυνατότητα όμως αυτή αρχικά απαιτούταν είτε η εγκατάσταση κάποιου προγράμματος όπως το skype ή η εγκατάσταση κάποιου προσθέτου στον φυλλομετρητή (google hangouts). Με σκοπό λοιπόν να γίνει πιο εύκολη η πρόσβαση σε αυτές τις υπηρεσίες, αναπτύχθηκαν τεχνολογίες που αφαιρούν την ανάγκη κάποιας επιπλέον εγκατάστασης από τη μεριά των χρηστών. Το μόνο που χρειάζεται πλέον είναι ο φυλλομετρητής που χρησιμοποιεί για την περιήγησή του στο διαδίκτυο. Το σύνολο αυτών των τεχνολογιών ονομάζεται WebRTC και υποστηρίζεται από όλους τους δημοφιλείς φυλλομετρητές. Στην παρούσα διπλωματική καλούμαστε να αναπτύξουμε μια web εφαρμογή στην οποία οι χρήστες συνδέονται με μοναδικό προαπαιτούμενο τον φυλλομετρητή τους και τους δίνεται η δυνατότητα να πραγματοποιούν κλήσεις ήχου και εικόνας μεταξύ δύο ή και περισσοτέρων συμμετεχόντων. Επιπρόσθετα, εφόσον το επιθυμούν, έχουν την δυνατότητα να καταγράψουν την συνομιλία τους η οποία στη συνέχεια είναι άμεσα διαθέσιμη για αναπαραγωγή. Η επικοινωνία γίνεται εφικτή με τη χρήση του πρωτοκόλλου WebRTC. Για την ανάπτυξη της εφαρμογής χρησιμοποιήθηκε το περιβάλλον nodejs και για τη βάση δεδομένων του συστήματος η MongoDB. Κεντρικό ρόλο στην καταγραφή της επικοινωνίας παίζει ο διακομιστής πολυμέσων Kurento.

Λέξεις κλειδιά

Πολλαπλή επικοινωνία, καταγραφή επικοινωνίας, εφαρμογή ιστού, mongoDB, Kurento, nodeJS, Javascript, CSS, HTML5, WebRTC

Abstract

Along with the evolution of internet and its gradual expansion in everyday life of people, it is being used more and more as a means of direct communication. Its use gives us also the ability to add video to the communication. But for this ability, in the beginning, it was required either to install a program like skype or to install an addon to the browser (google hangouts). So having as a goal to make it easier to access these services, there have been developed technologies that remove the requirement of an additional installation from the user. The only requirement is now the browser that is being used for browsing the internet. All these technologies together form the WebRTC protocol which is being supported from all the major browsers. In this diploma thesis we are called to develop a web application to which users connect, having their browser as the only requirement, and they have the ability to make video calls between two or more participants. Moreover, if they wish, they can record their communication and have the recording promptly available for playback. The communication becomes possible using the WebRTC protocol. NodeJS environment was used for the development of the web application and mongoDB for the database. Kurento Media Server plays a central role in the recording of the communication.

Keywords

Multipoint communication, recording communication, web application, mongoDB, Kurento, nodeJS, Javascript, CSS, HTML5, WebRTC

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τους καθηγητές κ. Παναγιώτη Τσανάκα και κ. Ηλία Μαγκλογιάννη για τις πολύτιμες συμβουλές και την εμπιστοσύνη που έδειξαν προς το πρόσωπό μου αναθέτοντάς μου την παρούσα διπλωματική εργασία.
Επίσης, ευχαριστώ θερμά τους φίλους μου για την ηθική και όχι μόνο συνεισφορά τους καθ' όλη την διάρκεια των σπουδών μου.
Τέλος θέλω να ευχαριστήσω τους γονείς και τον αδερφό μου για την κατανόηση, την υποστήριξη και τις συμβουλές τους.

Περιεχόμενα

Περιεχόμενα Εικόνων

1. Εισαγωγή

1.1. Πρόβλημα

Στη σύγχρονη κοινωνία, το Διαδίκτυο είναι μέρος της καθημερινής μας ζωής. Μπορούμε να το βρούμε παντού, όπως για παράδειγμα σε σπίτια, καφετέριες, λεωφορεία ακόμη και στα κινητά τηλέφωνα μας. Το διαδίκτυο έχει δημιουργήσει έναν ολοκαίνουργιο τρόπο επικοινωνίας μεταξύ των ανθρώπων σε ολόκληρο τον κόσμο, αφού είναι διαθέσιμο παντού.

Από την άλλη, το πρόγραμμα περιήγησης στο διαδίκτυο έχει γίνει μία από τις πιο συχνά χρησιμοποιούμενες εφαρμογές σε όλες σχεδόν τις συσκευές. Με άλλα λόγια, είναι ένα αναπόσπαστο κομμάτι της ζωής μας. Για παράδειγμα ορισμένοι χρήστες υπολογιστών δεν χρησιμοποιούν καμία άλλη εφαρμογή πέρα από το πρόγραμμα περιήγησης.

Το Διαδίκτυο αρχικά χρησίμευε για τη δημοσίευση στατικού περιεχομένου. Ωστόσο, εφαρμογές πλούσιου περιεχομένου έχουν αρχίσει επίσης να εμφανίζονται σε αυτό. Αυτές οι εφαρμογές μοιάζουν κάποιες φορές με παλιομοδίτικες εφαρμογές desktop αλλά παρέχουν ιστοσελίδες με διαδραστικότητα, όπως για παράδειγμα κινούμενα αντικείμενα στη σελίδα, καλύτερα γραφικά, ασύγχρονη επαναφόρτωση πληροφοριών και αναπαραγωγή πολυμέσων.

Επίσης, εφαρμογές πραγματικού χρόνου έχουν αρχίσει να επεκτείνονται. Τέτοιες εφαρμογές καθιστούν δυνατή την επικοινωνία μεταξύ client και server. Πιο συγκεκριμένα, οι clients μπορούν να στέλνουν αιτήματα στο server, ενώ ο server μπορεί να παρέχει σε όλους τους συνδεδεμένους clients νέες πληροφορίες μόλις αυτές είναι διαθέσιμες. Τέτοιες τεχνολογίες πραγματικού χρόνου καθιστούν δυνατή τη δημιουργία εφαρμογών επικοινωνίας πολυμέσων απευθείας στο πρόγραμμα περιήγησης. Οι εφαρμογές αυτές χρησιμοποιούνται κυρίως στα κοινωνικά μέσα δικτύωσης ή σε διαδικτυακά παιχνίδια, όπου η ανανέωση των δεδομένων σε πραγματικό χρόνο είναι πολύ σημαντική.

Στο παρελθόν, για να είναι εφικτή η ενημέρωση περιεχομένου σε εφαρμογές σε πραγματικό χρόνο, χρησιμοποιούνταν κυρίως plugins. Για να είναι δυνατόν αυτά να

χρησιμοποιηθούν, θα έπρεπε αρχικά να προστεθούν και να εγκατασταθούν στο εκάστοτε πρόγραμμα περιήγησης. Αυτή η λύση δεν είναι ιδανική, καθώς οι χρήστες των προγραμμάτων περιήγησης, θα πρέπει να τα εγκαθιστούν και στη συνέχεια να τα συντηρούν. Για παράδειγμα, θα έπρεπε να ψάχνουν για ενημερώσεις μετά την εγκατάστασή τους. Αυτό είναι ένα δύσκολο κομμάτι, για ορισμένους χρήστες που δεν έχουν τις απαραίτητες γνώσεις. Επιπλέον κάποια από αυτά τα plugins, αφήνουν κενά στην ασφάλεια του υπολογιστή του χρήστη. Παραδείγματα τέτοιων plugins είναι το Adobe Flash και το Silverlight. Τέλος, είναι δύσκολη από έναν μη γνώστη του αντικειμένου η λύση σχετικών προβλημάτων που μπορούν να εμφανιστούν.

Τελικά, η χρήση των plugins αυτών μπορεί να αντικατασταθεί πλέον από τα HTML5 και Javascript, τα οποία έχουν εξαπλωθεί αρκετά, προσφέροντας νέες δυνατότητες στους προγραμματιστές. Αυτές οι τεχνολογίες δίνουν τη δυνατότητα δημιουργίας διαδραστικών εφαρμογών χωρίς να απαιτείται η προσθήκη plugins στο πρόγραμμα περιήγησης. Για να το πετύχουν αυτό, χρησιμοποιούν εγγενείς δυνατότητες του προγράμματος περιήγησης.

Μία από τις πιο πρόσφατες προκλήσεις στον τομέα της επικοινωνίας, είναι η ανθρώπινη επικοινωνία μέσω φωνής και βίντεο σε πραγματικό χρόνο (Real Time Communication, RTC). Τέτοιες εφαρμογές είναι απαραίτητο να βρίσκονται σε εφαρμογές Ιστού και θα πρέπει να είναι τόσο απλές όσο και η εισαγωγή κειμένου σε ένα πεδίο κειμένου. Χωρίς αυτές, δεν είναι δυνατόν να καινοτομούμε και να αναπτύσσουμε νέους τρόπους αλληλεπίδρασης μεταξύ των ανθρώπων.

Πάντα η επικοινωνία σε πραγματικό χρόνο ήταν μια αρκετά απαιτητική και πολύπλοκη τεχνολογία, η οποία συνδυάζονταν με άλλες ακριβές τεχνολογίες όπως αυτή του ήχου και του βίντεο. Αυτό γίνεται ακόμα πιο απαιτητικό, όταν η τεχνολογία αυτή πρέπει να ενσωματωθεί στον ιστό.

Το Μάιο του 2011, η Google κατάφερε να παρουσιάσει το WebRTC, που είχε όλες τις δυνατότητες που αναφέραμε παραπάνω. Πιο συγκεκριμένα, ενσωμάτωσε όλες τις νέες τεχνολογίες και υπηρεσίες της σε ένα open source project για περιηγητές βασισμένο σε επικοινωνία πραγματικού χρόνου με τη χρήση βίντεο και ήχου, με το όνομα WebRTC. Επιπλέον, συνεργάστηκε με τις κοινότητες προτύπων τεχνολογιών

IETF και W3C για να καθορίσει και να εφαρμόσει ένα σύνολο προτύπων για επικοινωνίες πραγματικού χρόνου. Το W3C draft του WebRTC είναι μία δουλειά που βρίσκεται ακόμα σε εξέλιξη με προηγμένες εφαρμογές για τους browsers Chrome και Mozilla. Το συγκεκριμένο API βασίστηκε σε μία προηγούμενη δουλειά, που είχε γίνει στο WHATWG και αναφερόταν ως ConnectionPeer API, καθώς επίσης, και σε μία ιδέα που είχε αναπτυχτεί στα εργαστήρια της Ericsson χωρίς όμως να έχει γίνει προτυποποίηση. Το WebRTC από την αρχή είχε την ενεργή υποστήριξη από πολλούς οργανισμούς όπως για παράδειγμα των Mozilla, Cisco και Opera.

1.2. Στόχοι

Αυτή η εργασία ασχολείται με τη δημιουργία μίας διαδικτυακής πύλης η οποία θα χρησιμοποιηθεί για την επικοινωνία γιατρού – ασθενή σε πραγματικό χρόνο με τη χρήση ήχου και βίντεο. Τα περισσότερα σημερινά αντίστοιχα συστήματα έχουν τη μορφή επιτραπέζιων εφαρμογών. Και αυτό σημαίνει ότι ο χρήστης είναι υποχρεωμένος να κατεβάσει, να εγκαταστήσει και να συντηρεί την αντίστοιχη εφαρμογή στον υπολογιστή του. Επιπλέον, η χρήση τέτοιων εφαρμογών μπορεί να μην είναι δωρεάν, και να απαιτείται συνδρομή για αριθμό συμμετεχόντων μεγαλύτερο των δύο ατόμων. Το πρόγραμμα που θα παρουσιαστεί στη συνέχεια βασίζεται στην τεχνολογία WebRTC.

Η τεχνολογία WebRTC παρέχει διασυνδέσεις για επικοινωνία σε πραγματικό χρόνο, απευθείας, χρησιμοποιώντας μόνο ένα πρόγραμμα περιήγησης. Παρέχει λειτουργίες για την αποστολή ήχου, βίντεο και αρχείων. Ενώ ταυτόχρονα, η διεπαφή αυτή μπορεί να είναι μέρος σχεδόν όλων των φυλλομετρητών. Κατ' επέκταση, δεν υπάρχει η απαίτηση από τον χρήστη εγκατάστασης κάποιας εφαρμογής ή plugin για να χρησιμοποιήσει αυτήν την πύλη. Στην πραγματικότητα, το WebRTC είναι διαθέσιμο σε όλα τα προγράμματα περιήγησης, σε όλους τους χρήστες ανεξάρτητα από το ποια συσκευή μπορεί να χρησιμοποιούν. Επιπλέον, το WebRTC, όπως θα δούμε και στη συνέχεια, δίνει τη δυνατότητα επιλογής των χρηστών που θα συμμετέχουν σε μία συγκεκριμένη συνδιάσκεψη.

Το θεωρητικό μέρος αυτής της εργασίας σχετίζεται με την περιγραφή και την ανάλυση των διαθέσιμων τεχνολογιών που μπορούν να χρησιμοποιηθούν για τη δημιουργία εφαρμογών Internet πλούσιου περιεχομένου. Το μεγαλύτερο θεωρητικό μέρος σχετίζεται με την περιγραφή της τεχνολογίας WebRTC και διεπαφές που αυτό παρέχει στους προγραμματιστές. Ενώ, στο πρακτικό μέρος περιγράφεται η υλοποίηση της πύλης με τη χρήση παραδειγμάτων.

Αναλυτικότερα, το δεύτερο κεφάλαιο αποτελεί μία επέκταση του παρόντος κεφαλαίου, αφού συνεχίζει να αναλύει σε βάθος τους στόχους της συγκεκριμένης εργασίας και εστιάζει σε ζητήματα επικοινωνίας μεταξύ των χρηστών. Πιο συγκεκριμένα, αρχικά αναφέρεται στην πολλαπλή επικοινωνία με τη χρήση του πρωτοκόλλου WebRTC. Στη συνέχεια αναλύει τη δυνατότητα καταγραφής που δίνεται από το πρωτόκολλο WebRTC και τέλος αναφέρεται στους πιθανούς περιορισμούς του δικτύου που μπορούν να επηρεάσουν το αποτέλεσμα, δηλαδή την επικοινωνία μεταξύ γιατρού – ασθενή.

Στο κεφάλαιο τρία, αναφέρονται και αναλύονται όλες οι τεχνολογίες που χρησιμοποιήθηκαν για την ανάπτυξη του συγκεκριμένου συστήματος. Έτσι, αναφερόμαστε τόσο σε τεχνολογίες διαδικτύου όπως οι HTML, CSS, Javascript κ.α., όσο και σε πρωτόκολλα όπως το NAT και το ICE. Επιπλέον, αναφερόμαστε σε άλλες τεχνολογίες όπως το Kurento που προσφέρει προχωρημένες δυνατότητες επεξεργασίας δεδομένων όπως το video και ο ήχος, το MongoDB, τη βάση δεδομένων που χρησιμοποιείται στο σύστημα για την αποθήκευση δεδομένων και τέλος το NodeJS.

Στο τέταρτο κεφάλαιο αναφερόμαστε στην αρχιτεκτονική που έχει ακολουθηθεί κατά την υλοποίηση του συστήματός μας. Χωρίζουμε το κεφάλαιο αυτό σε δύο επιμέρους υποενότητες, όπου στην πρώτη αναλύεται η πλευρά του server, ή εξυπηρετητή, και στην δεύτερη η πλευρά του client. Πιο συγκεκριμένα στην πλευρά του εξυπηρετητή, αναλύονται σε βάθος τα Kurento, Signaling και MongoDB. Στην πλευρά του client παρατίθενται τα APIs που προσφέρει το WebRTC.

Στο κεφάλαιο πέντε παρουσιάζεται αναλυτικά το σύστημα που έχει αναπτυχθεί. Αναλύονται όλες οι διαθέσιμες λειτουργίες ενώ δίνονται στιγμιότυπα οθόνης σαν παραδείγματα για κάθε μία από αυτές. Στη συνέχεια αναφερόμαστε σε όλα τα

μηνύματα που ανταλλάσσονται μεταξύ client και server και τις αντίστοιχες συναρτήσεις που καλούνται για να είναι δυνατή η επικοινωνία μεταξύ τους.

Τέλος, στο έκτο κεφάλαιο παρατίθενται τα συμπεράσματα στα οποία καταλήξαμε μέσα από αυτή την εργασία. Τα συμπεράσματα σχετίζονται κυρίως με το σύστημα που αναπτύχθηκε. Παράλληλα συνοψίζουμε τις δυνατότητες που προσφέρει το WebRTC τόσο στους απλούς χρήστες όσο και στους προγραμματιστές.

2. Ανάλυση στόχων

2.1. Πολλαπλή επικοινωνία

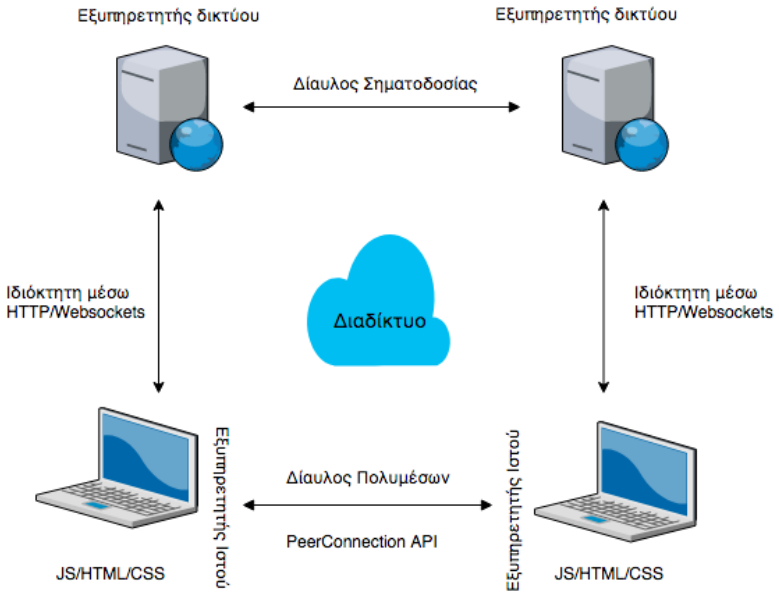
Η επικοινωνία σε πραγματικό χρόνο μέσω Web, με τη χρήση του WebRTC, αποτελεί μία καινοτόμα βιομηχανική προσπάθεια που επεκτείνει το μοντέλο περιήγησης που χρησιμοποιείται στο διαδίκτυο. Για πρώτη φορά, τα προγράμματα περιήγησης ιστού έχουν τη δυνατότητα άμεσης ανταλλαγής πληροφοριών σε πραγματικό χρόνο μεταξύ τους με τη χρήση της peer-to-peer λογικής.

Το World Wide Web Consortium (W3C) και το Internet Engineering Task Force (IETF) από κοινού όρισαν JavaScript APIs, τα tags της HTML5 και τα απαιτούμενα πρωτόκολλα επικοινωνίας για τη δημιουργία και τη διαχείριση ενός αξιόπιστου καναλιού επικοινωνίας μεταξύ οποιοδήποτε ζεύγος σύγχρονων προγραμμάτων περιήγησης ιστού.

Σκοπός τους ήταν η τυποποίηση ενός WebRTC API που θα είναι συμβατό με κάθε πρόγραμμα περιήγησης ιστού και θα επιτρέπει σε μια εφαρμογή web που εκτελείται σε μία οποιαδήποτε συσκευή, μέσω ασφαλούς πρόσβασης στις μονάδες εισόδου – εξόδου, όπως για παράδειγμα κάμερες και μικρόφωνα, να ανταλλάσσει πολυμέσα και δεδομένα, σε πραγματικό χρόνο, με μία απομακρυσμένη εφαρμογή web μέσω μιας peer- to-peer αρχιτεκτονικής επικοινωνίας.

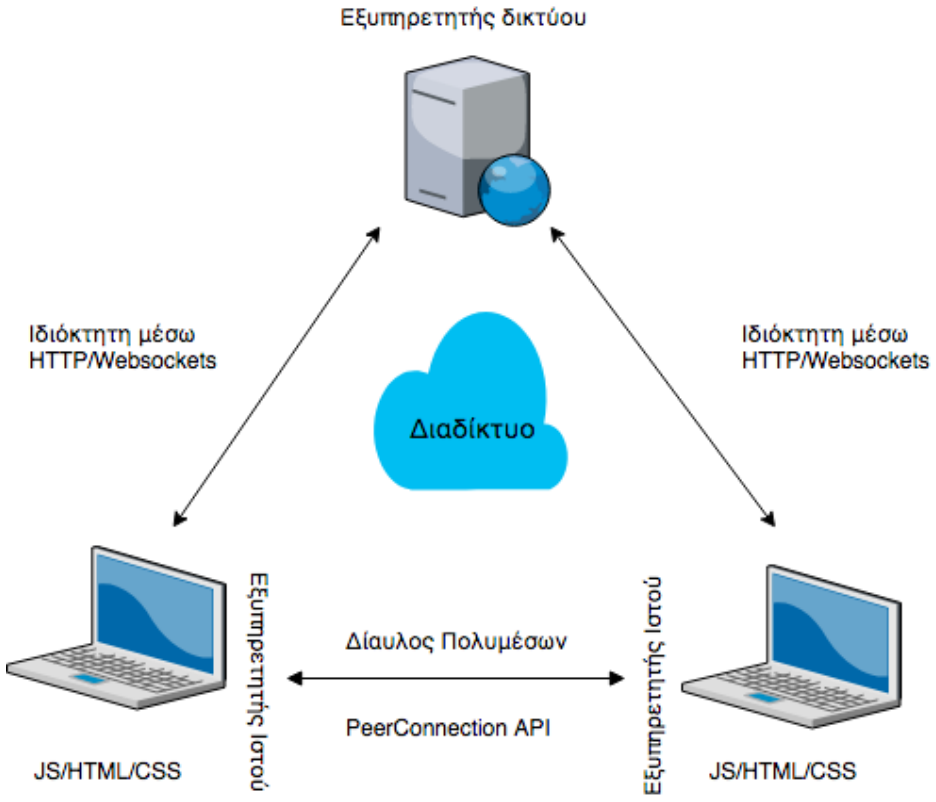
Το WebRTC API προβλέπει μια συνεχής ροή δεδομένων σε πραγματικό χρόνο, που μεταδίδεται μέσω του δικτύου. Έτσι, επιτρέπεται η απευθείας επικοινωνία μεταξύ δύο browsers, χωρίς περαιτέρω μεσάζοντες κατά μήκος της διαδρομής.

Το WebRTC επεκτείνει το μοντέλο client - server, που χρησιμοποιείται στον παγκόσμιο ιστό, εισάγοντας ένα μοντέλο peer-to-peer επικοινωνίας μεταξύ προγραμμάτων περιήγησης. Για να είναι κατανοητό το πως τελικά επιτυγχάνεται αυτή η επικοινωνία, παρακάτω δίνεται το γενικό αρχιτεκτονικό μοντέλο του WebRTC. Το μοντέλο αυτό έχει στηριχτεί στο SIP (Session Initiation Protocol) Trapezoid (RFC3261) πρωτόκολλο.



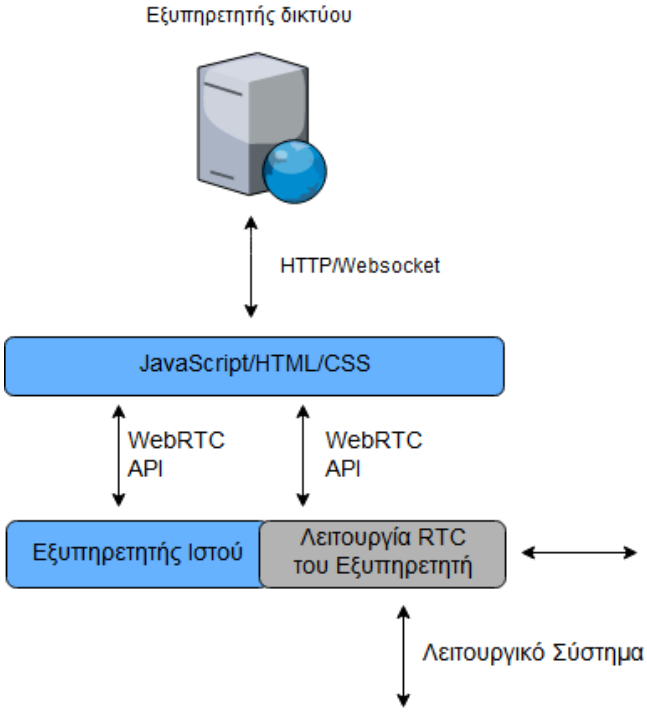
Εικόνα 1: Το WebRTC Trapezoid πρωτόκολλο.

Στο μοντέλο Trapezoid WebRTC, και οι δύο browsers εκτελούν μια εφαρμογή ιστού, η οποία έχει ληφθεί από διαφορετικούς servers. Η έναρξη και ο τερματισμός της επικοινωνίας πραγματοποιείται μέσω μηνυμάτων σηματοδοσίας, τα οποία μεταφέρονται μέσω των πρωτοκόλλων HTTP ή WebSocket και στέλνονται από εξυπηρετητές ιστού που μπορούν να τροποποιήσουν, να μεταφράσουν ή να διαχειριστούν τα μηνύματα αυτά, όπως απαιτείται. Η σηματοδοσία που χρησιμοποιείται μεταξύ του browser και του server δεν είναι τυποποιημένη από το WebRTC, και θεωρείται μέρος της εφαρμογής. Η ανταλλαγή δεδομένων μεταξύ των browsers των χρηστών γίνεται με τη χρήση PeerConnection, χωρίς την ύπαρξη παρεμβαλλόμενων διακομιστών. Τέλος, οι δύο διακομιστές ιστού μπορούν να επικοινωνούν χρησιμοποιώντας ένα πρότυπο πρωτόκολλο σηματοδοσίας όπως τα SIP ή Jingle (XEP-0166). Ωστόσο, μπορεί να χρησιμοποιηθεί και ένα πρωτόκολλο σηματοδότησης που έχει αναπτυχθεί από την εφαρμογή. Το πιο συνηθισμένο χρησιμοποιούμενο σενάριο WebRTC είναι αυτό που και οι δύο browsers τρέχουν την ίδια εφαρμογή ιστού, η οποία λαμβάνεται από τον ίδιο server. Στην περίπτωση αυτή το τραπεζοειδές σχήμα γίνεται ένα τρίγωνο, όπως φαίνεται στην εικόνα 2.



Εικόνα 2: The WebRTC Triangle πρωτόκολλο.

Μια εφαρμογή ιστού WebRTC, συνήθως είναι γραμμένη σαν ένα συνδυασμό HTML και JavaScript, και αλληλεπιδρά με τους περιηγητές ιστού μέσω του τυποποιημένου API WebRTC. Έτσι είναι δυνατή η λειτουργία και ο έλεγχος της σωστής λειτουργίας του browser σε πραγματικό χρόνο. Η εφαρμογή ιστού WebRTC μπορεί επίσης, να αλληλεπιδρά με τον browser, χρησιμοποιώντας και τα δύο WebRTC και άλλα τυποποιημένα API. Σκοπός είναι τόσο η διερεύνηση των δυνατοτήτων του προγράμματος περιήγησης όσο και η λήψη ειδοποιήσεων από το πρόγραμμα περιήγησης.



Εικόνα 3: Επικοινωνία σε πραγματικό χρόνο στο πρόγραμμα περιήγησης

Για να είναι όλα τα παραπάνω εφικτά, το API του WebRTC θα πρέπει να παρέχει ένα ευρύ φάσμα λειτουργιών, όπως διαχείριση peer-to-peer συνδέσεων, διαπραγμάτευση δυνατοτήτων κωδικοποίησης / αποκωδικοποίησης, επιλογή και έλεγχο των media, του firewall / τείχους προστασίας και των στοιχείων NAT.

2.2. Καταγραφή

Όπως έχουμε ήδη αναφέρει, το WebRTC χρησιμοποιείται σε κλήσεις φωνής / video, τηλεδιασκέψεις και κοινή χρήση αρχείων P2P. Ωστόσο προσφέρει τη δυνατότητα καταγραφής ροής πολυμέσων, όπως το video και οι φωνητικές κλήσεις. Σε γενικές γραμμές, υπάρχουν 3 διαφορετικοί μηχανισμοί που μπορούν να χρησιμοποιηθούν για καταγραφή:

- Εγγραφή από την πλευρά του server (Server side recording)
- Εγγραφή στην πλευρά του client (Client side recording)
- Μετάδοση μέσων (Media Forwarding)

Στο επόμενο κομμάτι θα αναλύσουμε αυτές τις πιθανές περιπτώσεις καταγραφής.

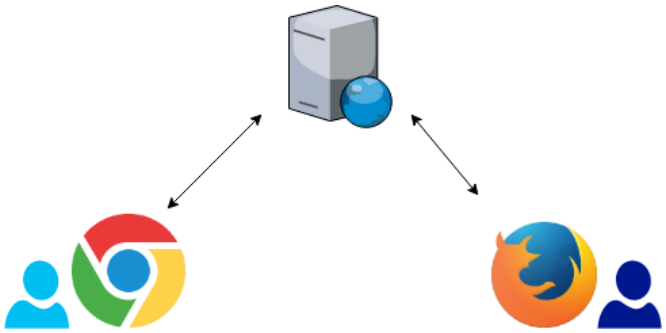
2.2.1. Εγγραφή από την πλευρά του server

Αυτή η τεχνική χρησιμοποιείται συνήθως από τους προγραμματιστές και ταιριάζει καλύτερα στις περισσότερες περιπτώσεις. Σε αυτή την περίπτωση, απαιτείται η χρήση ενός ενδιάμεσου διακομιστή πολυμέσων (media server) και η δρομολόγηση των μέσων ενημέρωσης μέσω αυτού αντί απευθείας μεταξύ των browsers.

Ο ενδιάμεσος διακομιστής πολυμέσων έχει την δυνατότητα να «διαβάσει» το περιεχόμενο των πακέτων καθώς αυτά είναι κρυπτογραφημένα μόνο κατά την μετάδοση τους από το ένα άκρο στο άλλο. Αυτό που γίνεται, στην πραγματικότητα, είναι ο τερματισμός των WebRTC session στον ενδιάμεσο διακομιστή πολυμέσων και από τις δύο πλευρές της κλήσης. Πιο συγκεκριμένα, τα μέσα κατευθύνονται μέσω του διακομιστή και ταυτόχρονα στέλνονται εκ νέου κωδικοποιημένα μετά από επεξεργασία και καταγραφή.

Ένας διακομιστής πολυμέσων WebRTC λειτουργεί σαν ένα ενδιάμεσο επίπεδο πολυμέσων, απ’ όπου περνάει όλη η κίνηση των μέσων ενημέρωσης, καθώς κινείται από την πηγή στον προορισμό. Ο διακομιστής πολυμέσων μπορεί να βρίσκεται οπουδήποτε στο δίκτυο και δεν αντιπροσωπεύει έναν από τους συμμετέχοντες στην συνομιλία αλλά μια ειδική οντότητα που εκτελεί λειτουργίες προκειμένου να παρέχει προηγμένες υπηρεσίες επικοινωνίας. Επιπροσθέτως, σκοπός του διακομιστή πολυμέσων είναι, μεταξύ άλλων, να ξεπεραστούν όλοι οι πιθανοί περιορισμοί δικτύου, όπως θα δούμε στη συνέχεια. Στην απλούστερη περίπτωση, ο διακομιστής πολυμέσων πρέπει μόνο να ανακατευθύνει ροές μέσων μεταξύ των συμμετεχόντων, ώστε να μπορούν να λαμβάνονται δεδομένα από άλλους χρήστες, δρώντας ως το κέντρο μιας τοπολογίας αστέρα.

Συνήθως, ο διακομιστής πολυμέσων είναι ένα κομμάτι λογισμικού που μπορεί να επικοινωνεί απρόσκοπτα με όλους τα συνδεδεμένους WebRTC clients, ενώ ταυτόχρονα έχει τη δυνατότητα να ανακατευθύνει και να μεταφράζει ροές πολυμέσων κατά βούληση.



Εικόνα 4: Δρομολόγηση των δεδομένων μέσω media server

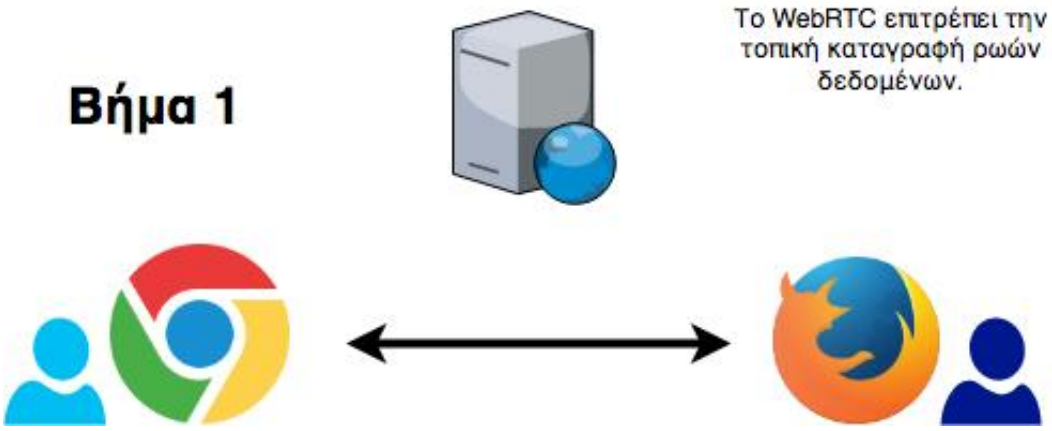
Ο διακομιστής πολυμέσων έχει τη δυνατότητα να επεξεργάζεται εισερχόμενες ροές πολυμέσων και να προσφέρει διάφορες υπηρεσίες, όπως αυτή της καταγραφής. Με τον όρο "καταγραφή", εννοούμε πως τα μεταδιδόμενα μέσα μεταξύ των συνομιλητών μπορούν να καταγράφονται και να αποθηκεύονται με μόνιμο τρόπο για οποιονδήποτε σκοπό.

Η επεξεργασία των δεδομένων στον ενδιάμεσο διακομιστή πολυμέσων μπορεί να περιλαμβάνει την ανάμιξη όλων των εισερχόμενων ροών δεδομένων από όλους τους συμμετέχοντες και τον συνδυασμό τους σε ένα ενιαίο αρχείο πολυμέσων, το οποίο στη συνέχεια στέλνεται στους τελικούς χρήστες (multiplexing of data). Επιπλέον, έχει τη δυνατότητα μείωσης του μεγέθους του αρχείου που στέλνεται και τελικά αποθηκεύεται (file compression). Και τέλος, μπορεί να αλλάζει ο τύπος (format) του αρχείου σε κάποια επιθυμητή μορφή, για να είναι δυνατή η αναπαραγωγή του και σε άλλους τύπους συσκευών και μέσων (encoding και format change).

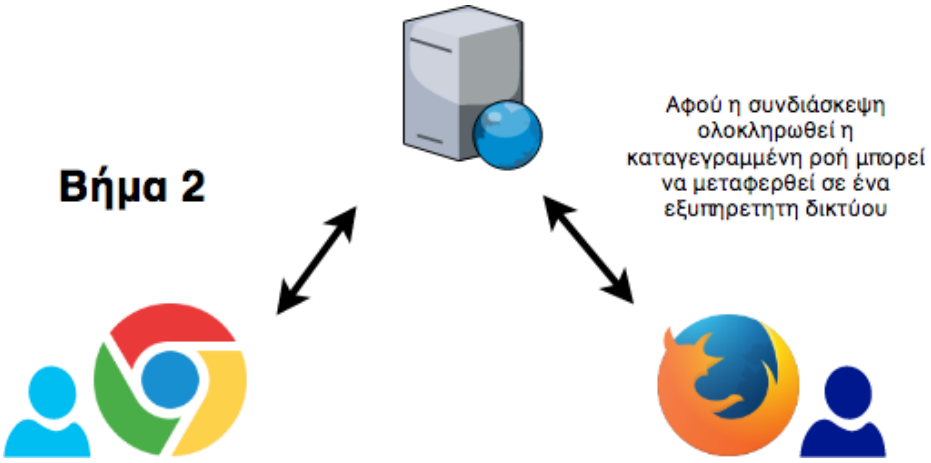
2.2.2. Εγγραφή στην πλευρά του client

Πέρα από την δυνατότητα καταγραφής στην πλευρά του server είναι δυνατή και η καταγραφή στην πλευρά του client. Αυτή η τεχνική είναι απλούστερη και προτιμάται όταν η ανάπτυξη του WebRTC γίνεται από προγραμματιστές Javascript και όχι full-stack προγραμματιστές μιας και δεν απαιτείται η ανάπτυξη ενός πολύπλοκου backend που θα υποστηρίζει τη δομή της εφαρμογής.

Η καταγραφή από τη πλευρά του client, απαιτεί δύο ξεχωριστά βήματα. Αρχικά, γίνεται καταγραφή του υλικού στον client, καθώς το WebRTC το επιτρέπει με τη χρήση του MediaStream Recording API. Και στη συνέχεια, ανεβάζουμε αυτό που έχει καταγραφεί στο διακομιστή. Σε αυτό το βήμα, δεν χρησιμοποιείται το WebRTC, απλά μετά φορτώνεται το αρχείο στο server. Στις ακόλουθες εικόνες φαίνονται τα δύο αυτά βήματα.



Εικόνα 5: Βήμα 1 - Καταγραφή των δεδομένων τοπικά



Εικόνα 6: Βήμα 2 – Ανέβασμα του αρχείου καταγραφής στον server.

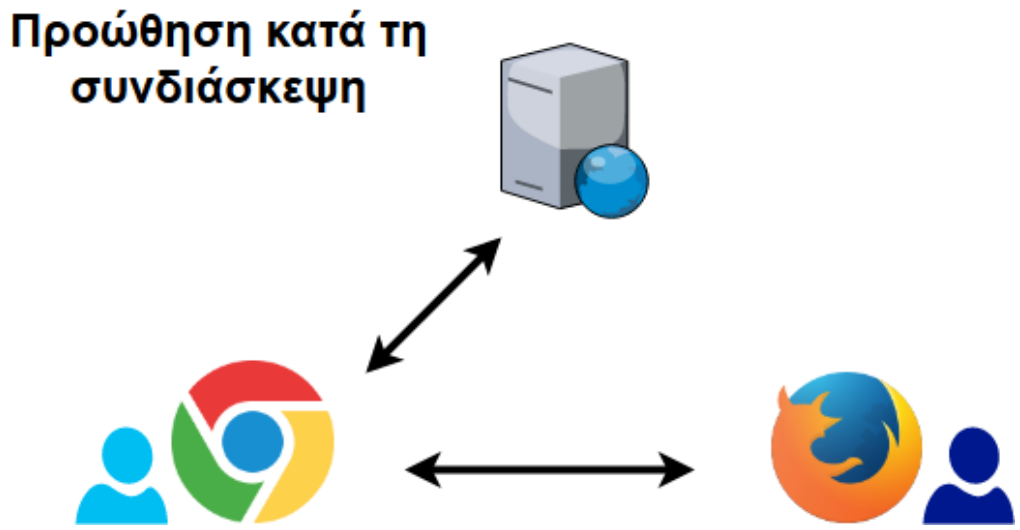
Ωστόσο, υπάρχουν ορισμένες προκλήσεις, που θα πρέπει να ληφθούν υπόψη σε αυτή την περίπτωση, καθώς η καταγραφή δεν μπορεί να ελεγχθεί μέσω του browser. Για

παράδειγμα, δεν είναι γνωστός ο διαθέσιμος αποθηκευτικός χώρος που μπορεί να χρησιμοποιηθεί για την εγγραφή. Αυτό μπορεί να είναι απαραίτητο ειδικά σε περιπτώσεις υπηρεσιών e-learning. Επιπλέον, ο χρήστης με την ολοκλήρωση του session, θα πρέπει να περιμένει για το ανέβασμα του αρχείου, το οποίο μπορεί να είναι αρκετά Gigabytes, χωρίς να κλείσει το πρόγραμμα περιήγησης ή την καρτέλα που φορτώνει την εγγραφή. Τέλος, ο χρήστης θα πρέπει να γνωρίζει τον τρόπο καταγραφής. Ενώ μπορεί να μην υπάρχει συγχρονισμός όταν και οι δύο πελάτες γράφουν ταυτόχρονα.

Από τα παραπάνω, γίνεται σαφές ότι η εγγραφή στην πλευρά του client, δεν είναι αποδοτική. Ωστόσο, θα μπορούσαν να γίνουν ενέργειες για να περιοριστούν ορισμένα από τα παραπάνω προβλήματα. Για παράδειγμα, μία λύση θα ήταν το σταδιακό ανέβασμα αποσπασμάτων καταγραφής κάθε λίγα δευτερόλεπτα ή λεπτά καθ 'όλη τη διάρκεια του session.

2.2.3. Μετάδοση μέσων

Πρόκειται για τη λιγότερο γνωστή τεχνική, η οποία χρησιμοποιείται λιγότερα από όλες. Ωστόσο, αποτελεί μια εναλλακτική λύση. Η τεχνική αυτή δεν επιτρέπει ούτε την τοπική καταγραφή του αρχείου ούτε την αποστολή του σε κάποιο απομακρυσμένο διακομιστή. Αυτό που γίνεται είναι η προώθηση των μέσων από τον έναν ή και τους δύο συμμετέχοντες σε ένα διακομιστή καταγραφής (recording server).



Εικόνα 7: Προώθηση μέσων σε έναν server καταγραφής κατά τη διάρκεια ενός session.

Σε αυτή την περίπτωση δηλαδή δημιουργείται μια επιπλέον σύνδεση κατά την διάρκεια της κλήσης προς έναν διακομιστή ο οποίος έχει το ρόλο του συμμετέχοντα με τη διαφορά πως μόνο λαμβάνει. Για να είναι αυτό δυνατόν, θα πρέπει να επιτρέπεται η μετάδοση των εισερχόμενων μέσων σύνδεσης μεταξύ των χρηστών. Το πλεονέκτημα σε αυτή την τεχνική είναι ότι δεν απαιτείται media server αλλά ένα εξωτερικός διακομιστής καταγραφής. Ο συμμετέχοντας όμως ο οποίος θα προωθεί τα πολυμέσα στο διακομιστή καταγραφής θα πρέπει να έχει το μεγαλύτερο εύρος ζώνης.

Τέλος, πέρα από την καταγραφή των δεδομένων, θα πρέπει να χρησιμοποιούνται επίσης μεταδεδομένα για την περαιτέρω επεξεργασία των μέσων. Για παράδειγμα, τα μεταδεδομένα είναι απαραίτητα για αναπαραγωγή, συγχρονισμό κ.λπ. Τέλος, εγείρονται θέματα ασφάλειας τόσο για την εγγραφή και την αναπαραγωγή.

2.3. Περιορισμοί δικτύου

Παρότι η επικοινωνία μεταξύ δύο χρηστών αποτελεί ήδη μια γνωστή τεχνολογία, η εφαρμογή του WebRTC, αναμένεται να έχει πολλές προκλήσεις, οι οποίες θα πρέπει να αντιμετωπιστούν. Ωστόσο, όλες οι προηγούμενες ανεπτυγμένες τεχνολογίες, παρέχουν υπόβαθρο και γνώση για να προβλέψουμε και να αντιμετωπίσουμε τις προκλήσεις αυτές. Οι προκλήσεις αυτές συνδέονται άμεσα με το περιβάλλον του

διαδικτύου όπου συμμετέχουν διαφορετικά είδη συσκευών. Οι συσκευές αυτές θα πρέπει να αλληλεπιδρούν με αποτελεσματικό τρόπο και να παρέχουν υψηλότερου επιπέδου υπηρεσίες, οι οποίες απαιτούνται σε περίπλοκα σενάρια, όπως για παράδειγμα, η εκπαίδευση ή η τηλεϊατρική. Στη συνέχεια θα παραθέσουμε αυτούς τους περιορισμούς δικτύου.

Ετερογενή δίκτυα πρόσβασης και συσκευές: Η ποικιλομορφία των συσκευών που έχουν πρόσβαση στον ιστό έχει αυξηθεί σημαντικά τα τελευταία χρόνια. Πιο συγκεκριμένα, οι διάφορες συσκευές που συνδέονται στον ιστό, έχουν διαφορετικές δυνατότητες και ίσως διαφορετικά δίκτυα πρόσβασης. Για παράδειγμα, χρησιμοποιούνται smartphones, τα οποία έχουν περιορισμένη επεξεργαστική ισχύς σε σύγκριση με τους προσωπικούς υπολογιστές, ενώ μπορεί να προσπελαύνουν τον ιστό μέσω κυψελοειδών δικτύων.

Μέγεθος οθόνης: Οι διαφορετικές συσκευές που προσπελαύνουν το διαδίκτυο, συνήθως έχουν και διαφορετικό μέγεθος οθόνης. Χαρακτηριστικό παράδειγμα αποτελούν τα smartphones, tablets και οι προσωπικοί υπολογιστές. Οι μικρότερες οθόνες δεν μπορούν να εμφανίσουν την ίδια ποσότητα πληροφορίας με τις μεγαλύτερες. Έτσι, σε μία τηλεδιάσκεψη, η πιθανή αποστολή ενός υψηλής ποιότητας video σε μια συσκευή μικρής οθόνης είναι ανούσια, καθώς οι χρήστες δεν μπορούν να παρατηρήσουν τη διαφορά. Επιπλέον, το μέγεθος της οθόνης περιορίζει τον αριθμό των συμμετεχόντων που μπορούν να προβληθούν ταυτόχρονα σε μία τηλεδιάσκεψη πολλαπλών συμμετεχόντων.

CPU: Η τηλεδιάσκεψη απαιτεί αρκετή επεξεργαστική ισχύ ώστε να είναι δυνατή η κωδικοποίηση, η αποκωδικοποίηση και η διανομή του βίντεο και ήχου σε πραγματικό χρόνο. Η ισχύς της CPU, που απαιτείται, εξαρτάται από πολλούς παράγοντες, όπως οι κωδικοποιητές που χρησιμοποιούνται, η ποιότητα του ήχου και του βίντεο και τέλος το μέγεθος του βίντεο. Στις κινητές συσκευές, η επεξεργαστική ισχύς είναι αρκετά περιορισμένη, ενώ η υπερβολική χρήση της συσκευής μπορεί να οδηγήσει σε μια αισθητή μείωση της διάρκειας ζωής της μπαταρίας.

Διαθεσιμότητα εύρους ζώνης και καθυστέρηση: Όπως αναφέρθηκε παραπάνω, η ποικιλία των χρησιμοποιούμενων συσκευών συμβαδίζει με μία ποικιλία

διαφορετικών δικτύων πρόσβασης. Για παράδειγμα, μπορεί να χρησιμοποιούνται τυπικά ενσύρματα δίκτυα Ethernet από προσωπικούς υπολογιστές και δίκτυα 3G και 4G από φορητές συσκευές. Όλες αυτές οι διαφορές θα πρέπει να λαμβάνονται υπόψη για τη βελτιστοποίηση της επικοινωνίας. Για παράδειγμα, σε συνδέσεις 3G, το διαθέσιμο εύρος ζώνης μπορεί να μεταβάλλεται χωρίς καμία προηγούμενη ειδοποίηση. Αυτό μπορεί να έχει σαν αποτέλεσμα, τη διακοπή της διάσκεψης εάν το σύστημα δεν είναι σε θέση να αντιδράσει αναλόγως.

Καταγραφή συνεδρίας: Σε ορισμένα σενάρια, όπως τα εταιρικά meeting και το e-learning, η καταγραφή όλων των πληροφοριών είναι ζωτικής σημασίας. Ωστόσο, το WebRTC, κατά την εγγραφή video, δεν παρέχει μηχανισμό για τη συλλογή και την αποθήκευση των streams όλων των συμμετεχόντων.

Gateway: Η λειτουργία και επικοινωνία διαφορετικών συστημάτων επικοινωνίας αποτελεί μία πρόκληση. Για παράδειγμα, η επικοινωνία μεταξύ SIP τηλεφώνων και των παραδοσιακών γραμμών τηλεφωνίας (Public Switched Telephone Networks - PSTN), επιτυγχάνεται με τη χρήση gateway που μεταφράζει τα σήματα σε ροές μέσων.

3. Τεχνολογικό Υπόβαθρο

3.1 HTML

Κατά βάση μια ιστοσελίδα στο διαδίκτυο είναι ένα έγγραφο html, αυτό το έγγραφο λαμβάνουν οι browsers και το χρησιμοποιούν για να απεικονίσουν τα περιεχόμενα της ιστοσελίδας. Η HTML είναι το ακρωνύμιο του αγγλικού **H**yper**T**ext **M**arkup **L**anguage (γλώσσα μορφοποίησης υπερκειμένου) και αποτελεί μια γλώσσα σήμανσης (markup language) η οποία περιγράφει τη δομής της ιστοσελίδας. Η HTML έχει δημιουργηθεί και συντηρείται από τον οργανισμό W3C.

Αναλύοντας την ονομασία της HTML βλέπουμε ότι αντικατοπτρίζει τον τρόπο με τον οποίο είναι δομημένες και λειτουργούν οι ιστοσελίδες στον παγκόσμιο ιστό. Πιο συγκεκριμένα η έννοια **H**yper**T**ext αναφέρεται στον τρόπο με τον οποίο μπορεί κάποιος να κινηθεί στο διαδίκτυο. Οι χρήστες μπορούν να ακολουθήσουν τα λεγόμενα **hyperlinks** μέσω των οποίων οδηγούνται σε άλλες ιστοσελίδες. Η έννοια **M**ark-up: αναφέρεται στα tags της HTML οι οποίες καθορίζουν τη μορφοποίηση του αντικειμένου που περικλείεται σε αυτά. Τα tags χρησιμεύουν για να ξεχωρίζουν τον κώδικα HTML από το κανονικό κείμενο. Τέλος ο όρος **L**anguage αναφέρεται στο γεγονός ότι η HTML περιλαμβάνει κωδικές λέξεις και ειδική σύνταξη όπως κάθε άλλη γλώσσα προγραμματισμού. Στην ακόλουθη εικόνα, φαίνεται η μορφή μίας HTML σελίδας.

Εικόνα 8: Μορφή της HTML[1]

Η HTML γράφεται χρησιμοποιώντας τα βασικά στοιχείων HTML τα οποία αποτελούνται από ετικέτες (tags), οι οποίες περικλείονται σε σύμβολα μεγαλύτερο από και μικρότερο από, για παράδειγμα <html>. Οι ετικέτες HTML συνήθως λειτουργούν ανά ζεύγη όπως για παράδειγμα οι ετικέτες <h1> και </h1> ενώ υπάρχουν και κάποιες ετικέτες οι οποίες λειτουργούν μόνες τους, μία τέτοια ετικέτα είναι η <img>. Στα ζεύγη ετικετών, η πρώτη ετικέτα ονομάζεται ετικέτα έναρξης (start tag) ή ετικέτα ανοίγματος και η δεύτερη ετικέτα λήξης (end tag) ή ετικέτα κλεισίματος αντίστοιχα. Ανάμεσα στις ετικέτες, οι σχεδιαστές ιστοσελίδων μπορούν να τοποθετήσουν κείμενο, πίνακες, εικόνες κλπ.

Η τελευταία της έκδοση είναι η HTML5 η οποία άρχισε να αναπτύσσεται από την ομάδα Web Hypertext Application Technology Working Group (WHATWG) τον Ιούνιο του 2004 και χρειάστηκαν δέκα χρόνια για να υιοθετηθεί ως το προτεινόμενο πρότυπο από την κοινοπραξία W3C (Web World Consortium), τον κύριο διεθνή οργανισμό προτύπων για το Web. Από εκεί και μετά τα πράγματα άρχισαν να κινούνται πιο γρήγορα αφού η μετάβαση από την έκδοση HTML 5.0 στην HTML 5.1 έγινε μόλις δύο χρόνια μετά, τον Νοέμβριο του 2016.

3.2 CSS

Η γλώσσα CSS (Cascading Style Sheet) δημιουργήθηκε και προτάθηκε από την κοινοπραξία W3C το 1996, ενώ ανήκει στην κατηγορία των style sheet γλωσσών

[1] https://commons.wikimedia.org/wiki/File:HTML.svg

προγραμματισμού. Χρησιμοποιείται για να παραμετροποιήσει την εμφάνιση εγγράφων που έχουν γραφτεί σε γλώσσες όπως οι HTML και XHTML. Έτσι, προσθέτει γνωρίσματα όπως το στυλ γραμματοσειράς, το χρώμα, οι διαστάσεις κτλ. Ένα στυλ μορφοποίησης μπορεί να αναφέρεται σε μια παράγραφο, σε έναν χαρακτήρα, σε ένα πίνακα και γενικά σε κάθε στοιχείο HTML. Αναθέτοντας μια κλάση σε μια ομάδα στοιχείων HTML μπορούμε να μεταβάλλουμε την εμφάνιση τους αλλάζοντας μία φορά τον κώδικα που αναφέρεται σε αυτή την κλάση. . Έχει αναπτυχθεί από τον οργανισμό W3C ο οποίος έχει δημιουργήσει και συντηρεί τις γλώσσες HTML και CSS. Ο ίδιος ο οργανισμός W3C ενθαρρύνει την χρήση της CSS αντί για διάφορα στοιχεία της HTML για σκοπούς παρουσίασης.

Εικόνα 9: Γλώσσα CSS[2]

Με τη χρήση της CSS προστέθηκε λειτουργικότητα και δυνατότητα σχεδιασμού στις ιστοσελίδες οι οποίες κάποτε ήταν απλά έγγραφα κειμένου. Στα πλεονεκτήματα της CSS συμπεριλαμβάνεται ο διαχωρισμός του περιεχομένου και της μορφοποίησης τα οποία βρίσκονται σε διαφορετικά αρχεία, η συνοχή της μορφοποίησης σε όλες τις σελίδες ενός ιστότοπου, η μείωση του όγκου δεδομένων που μεταφέρονται, η δυνατότητα αλλαγής με ευκολία ολόκληρης της μορφοποίησης του ιστότοπου αλλά και η δυνατότητα για διαφορετική μορφοποίηση της σελίδας ανάλογα με την συσκευή στην οποία εμφανίζεται και την εστίαση που επιλέγει ο χρήστης.

[2] https://www.wikihow.com/Define-a-CSS-Class-Style

Αυτός ο διαχωρισμός βελτιώνει την προσβασιμότητα του εγγράφου ενώ παρέχει μεγαλύτερη ευελιξία και έλεγχο στα διάφορα χαρακτηριστικά της εμφάνισης. Έτσι μπορούν πολλές διαφορετικές ιστοσελίδες να έχουν την ίδια μορφή, μειώνοντας την πολυπλοκότητα του κώδικα, αφού δεν χρειάζεται να επαναλαμβάνουμε πολλές φορές τα ίδια κομμάτια.

Εικόνα 10: Μορφοποίηση περιεχομένου με βάση το μέγεθος οθόνης[3]

3.3 Javascript

Η γλώσσα προγραμματισμού Javascript, μαζί με την HTML και τη CSS, είναι οι τρεις δομικές τεχνολογίες που απαρτίζουν την παραγωγή περιεχομένου του παγκόσμιου ιστού. Οι περισσότερες ιστοσελίδες την χρησιμοποιούν και όλοι οι σύγχρονοι web browsers (φυλλομετρητές) την υποστηρίζουν.

Η JavaScript είναι μια πρότυπη script γλώσσα προγραμματισμού. Για την ανάπτυξή της χρησιμοποιήθηκε το πρότυπο ECMAScript. Πρόκειται για μία client-side γλώσσα προγραμματισμού, η οποία χρησιμοποιείται κυρίως για να παρέχει πλούσιες διεπαφές στους χρήστες, έλεγχο δεδομένων (πριν την αποστολή μιας φόρμας στον server) ή διάφορα εφέ στις ιστοσελίδες. Επίσης παρέχει τα κατάλληλα εργαλεία ώστε να μπορεί κάποιος να διαχειριστεί διαδραστικά οποιοδήποτε στοιχείο της HTML, όπως είναι η επικεφαλίδα, οι πίνακες, οι φόρμες, τα κουμπιά και άλλα. Επιπρόσθετα μπορεί να εκτελεί υπολογισμούς, να αναλάβει την επικοινωνία με τον εξυπηρετητή

[3] http://www.digitalfamily.com/tutorials/how-to-create-responsive-adaptive-web-sites

και γενικά οποιαδήποτε λειτουργία θα περίμενε κανείς από μια γλώσσα προγραμματισμού. Με τον όρο client-side αναφερόμαστε στον κώδικα που εκτελείται από τη μεριά του χρήστη και όχι του εξυπηρετητή.

Η Javascript δίνει δυνατότητα αλληλεπίδρασης του χρήστη με εφαρμογές δημιουργώντας σελίδες με δυναμικό και αλληλεπιδραστικό περιεχόμενο. Λόγω του ότι είναι client-side γλώσσα, ο φυλλομετρητής (browser) του χρήστη φορτώνει όλο τον κώδικα και τον εκτελεί παρουσιάζοντας το αποτέλεσμα της εκτέλεσης. Με την εκτέλεση των <script> αντικειμένων στην πλευρά του χρήστη μειώνεται ο επεξεργαστικός φόρτος του εξυπηρετητή. Επειδή όμως η JavaScript είναι client-side, μπορεί ο χρήστης που περιηγείται σε μια ιστοσελίδα να μην έχει εγκατεστημένη την JavaScript ή να την έχει απενεργοποιημένη. Για αυτό ποτέ δεν πρέπει να βασίζεται ένας προγραμματιστής στην JavaScript για έλεγχο εγκυρότητας των δεδομένων που συμπληρώνονται σε μία φόρμα. Αυτό θα πρέπει να γίνεται σε συνδυασμό με ελέγχους από μία server-side γλώσσα προγραμματισμού.

3.4 Browsers

Με τον όρο browser ή web browser αναφερόμαστε σε ένα πρόγραμμα περιήγησης ιστού και αποτελεί μια εφαρμογή λογισμικού για την ανάκτηση, την παρουσίαση και τον διαμοιρασμό πληροφοριών στον Παγκόσμιο Ιστό. Μία πηγή πληροφορίας προσδιορίζεται μοναδικά από ένα αναγνωριστικό, που ονομάζεται Uniform Resource Identifier (URI / URL), ενώ μπορεί να είναι μια ιστοσελίδα, μία εικόνα, ένα βίντεο ή οποιοδήποτε άλλο περιεχόμενο. Εντός των ιστοσελίδων συχνά υπάρχουν υπερσύνδεσμοι (hyperlinks) που δίνουν τη δυνατότητα στους χρήστες να περιηγούνται εύκολα σε πληροφορίες με σχετικούς πόρους. Οι πιο δημοφιλής browers είναι οι Chrome, Internet Explorer, Safari, Opera και Firefox.

Παρόλο που οι browsers προορίζονται κυρίως για τη χρήση του Παγκόσμιου Ιστού, μπορούν επίσης να χρησιμοποιηθούν για την πρόσβαση σε πληροφορίες που παρέχονται από διακομιστές ιστού σε ιδιωτικά δίκτυα ή αρχεία σε συστήματα αρχείων.

Σκοπός των browsers είναι η προσπέλαση πληροφοριών από τον χρήστη («retrieval», «fetching»), η προβολή τους («display», «rendering») και στη συνέχεια η πρόσβαση άλλων πληροφοριών («navigation», «following links»).

Αυτή η διαδικασία ξεκινά όταν ο χρήστης εισάγει στο πρόγραμμα περιήγησης το ζητούμενο URL. Παράδειγμα ενός URL είναι το http://www.google.gr. Το πρώτο μέρος των URL, http:, προσδιορίζει τον τρόπο με τον οποίο θα ερμηνευτεί η διεύθυνση URL καθώς και τον πόρο από τον οποίο θα ανακτηθεί η πληροφορία. Έτσι, το http: προσδιορίζει έναν πόρο που θα ανακτηθεί μέσω του HTTP. Διάφορα άλλα προθέματα είναι τα *https:*, για ασφαλή σύνδεση, *ftp:* για τη μεταφορά αρχείων και *file:* για τοπικά αρχεία.

Στη συνέχεια όταν ο browser ανακτήσει την πληροφορία θα την εμφανίσει στην οθόνη του χρήστη. Για να είναι αυτό δυνατόν, όλο το περιεχόμενο μίας σελίδας, όπως το HTML και CSS, μετατρέπονται μέσω του layout engine, που διαθέτει ο browser, σε ένα διαδραστικό έγγραφο, που είναι αναγνώσιμο από τον τελικό χρήστη. Η διαδικασία αυτή χαρακτηρίζεται σαν rendering. Πιο συγκεκριμένα, ένας browser μπορεί να εμφανίζει εικόνες, αρχεία ήχου, βίντεο, XML αρχεία καθώς και εφαρμογές Flash και Java. Σε περιπτώσεις αρχείων μη υποστηριζόμενου τύπου, ο browser προτείνει στον χρήστη να αποθηκεύσει το αρχείο αυτό.

3.5 WebRTC

Το WebRTC είναι μια τεχνολογία που αναπτύχθηκε από την Google με στόχο να δώσει στους φυλλομετρητές (browsers) την δυνατότητα για επικοινωνία σε πραγματικό χρόνο (Real-Time Communication).

Ιστορικά η επικοινωνία πραγματικού χρόνου ήταν κάτι που αναπτυσσόταν από μεγάλες επιχειρήσεις τεχνολογίας που είτε χρησιμοποιούσαν ακριβές πατενταρισμένες τεχνολογίες ήχου και εικόνας είτε ανέπτυσσαν τις δικιές τους. Έτσι η ενσωμάτωση της RTC επικοινωνίας στις υπάρχουσες υπηρεσίες ήταν κάτι δύσκολο και αρκετά χρονοβόρο, ειδικά στο διαδίκτυο.

Η Google ενσωμάτωσε στην υπηρεσία email της την ζωντανή επικοινωνία ήχου και εικόνας με το Gmail Video chat το 2008 το οποίο απαιτούσε την εγκατάσταση του προσθέτου στον browser και στη συνέχεια κυκλοφόρησε το Hangouts το 2011. Βλέποντας λοιπόν την ανάγκη για μια κοινή πλατφόρμα ανάπτυξης εξαγόρασε την ίδια χρονιά την εταιρία GIPS η οποία είχε αναπτύξει αρκετά δομικά στοιχεία για την επικοινωνία RTC, όπως κωδικοποιητές ήχου βελτιωμένους για VoIP (Voice over IP). Στη συνέχεια διέθεσε στο κοινό την παραπάνω τεχνολογία ως υλικό ανοιχτού κώδικα με την ονομασία WebRTC. Για να επιτύχει την αποδοχή της βιομηχανίας συνεργάστηκε με τις αρμόδιες επιτροπές για την προτυποποίηση των πρωτοκόλλων, IETF και W3C. Η Ericsson Labs ήταν η πρώτη που διέθεσε μια υλοποίηση του WebRTC την οποία μπορούσαν χρησιμοποιήσουν οι developers.

Η τεχνολογία WebRTC είναι ένα σύνολο από Javascript APIs (διεπαφές) τα οποία επιτρέπουν σε web developers να παρέχουν υπηρεσίες επικοινωνίας στις σελίδες/εφαρμογές τους χωρίς την ανάγκη εγκατάστασης προσθέτων από την μεριά του χρήστη, το οποίο πολλές φορές λειτουργεί αποτρεπτικά, και χωρίς την κατ' επέκταση επιπλέον πολυπλοκότητα στην ανάπτυξη της υπηρεσίας.

Οι δομικές διεπαφές που γίνονται διαθέσιμες στους προγραμματιστές είναι

MediaStream API το οποίο σχεδιάστηκε για την εύκολη πρόσβαση σε ροές πολυμεσικών δεδομένων τα οποία προέρχονται από την κάμερα και το μικρόφωνο του χρήστη.

RTCPeerConnection API το οποίο είναι και η κυριότερη διεπαφή αφού ενθυλακώνει την δημιουργία, την διαχείριση και την κατάσταση της σύνδεσης μεταξύ των peers για την μεταφορά της ροής δεδομένων του καθενός, η οποία περιλαμβάνει βίντεο και ήχο.

RTCDataChannel API το οποίο δημιουργεί ένα αμφίδρομο κανάλι επικοινωνίας μεταξύ των peers το οποίο μπορούν να χρησιμοποιήσουν για να μεταδώσουν τυχαία δεδομένα (arbitrary data) όπως αρχεία και μηνύματα ή ακόμα και για τη μεταφορά δεδομένων με σκοπό τη δημιουργία αποκεντρωμένου δικτύου μεταξύ πολλών χρηστών.

3.6 Πρωτόκολλα (NAT, ICE, STUN, TURN)

3.6.1 NAT

Το πρωτόκολλο Network Address Translation (NAT) είναι μια μέθοδος χαρτογράφησης του χώρου διευθύνσεων IP. Πιο συγκεκριμένα, μπορεί να αντιστοιχεί μία διεύθυνση IP σε μία άλλη με την τροποποίηση των πληροφοριών διεύθυνσης δικτύου στην IP επικεφαλίδα των πακέτων που μπορεί να μεταφέρονται μέσω μιας συσκευή δρομολόγησης κίνησης. Είναι, δηλαδή, ένας μηχανισμός που επιτρέπει σε υπολογιστές που βρίσκονται πίσω από ένα δίκτυο και έχουν ιδιωτικές IPs να έχουν πρόσβαση στο internet, μεταφράζοντας την ιδιωτική IP σε δημόσια.

Η τεχνική χρησιμοποιήθηκε αρχικά για να αποφευχθεί η ανάγκη επαναπροσδιορισμού της υποδοχής ενός δικτύου όταν αυτό μεταφέρεται. Έτσι έγινε ένα δημοφιλές και σημαντικό εργαλείο για τη διατήρηση των IP διευθύνσεων στη μορφή IPv4, μέχρι αυτές να εξαντληθούν. Μια IP διεύθυνση μίας NAT πύλης μπορεί να χρησιμοποιηθεί για ένα ολόκληρο ιδιωτικό δίκτυο.

Η τεχνική IP masquerading «κρύβει» ολόκληρο το χώρο των διευθύνσεων IP, που συνήθως αποτελείται από ιδιωτικές διευθύνσεις IP, πίσω από μια ενιαία διεύθυνση IP σε ένα άλλο, συνήθως δημόσιο χώρο διευθύνσεων. Η διεύθυνση που πρέπει να κρύβεται αλλάζει σε μία ενιαία, κοινή διεύθυνση IP, η οποία είναι μία «νέα» διεύθυνση IP προέλευσης του εξερχόμενου πακέτου. Έτσι, φαίνεται ότι το πακέτο δεν προέρχεται από έναν ενιαίο host, αλλά από μία κοινή συσκευή. Λόγω της δημοτικότητας της τεχνικής αυτής για τη διατήρηση του IPv4 χώρου διευθύνσεων, το πρωτόκολλο NAT έχει γίνει σχεδόν συνώνυμο με τον όρο IP masquerading.

Όμως, η μετάφραση διευθύνσεων δικτύου που τροποποιεί τις πληροφορίες των διευθύνσεων IP στα πακέτα μπορεί να έχει σοβαρές επιπτώσεις στην ποιότητα της σύνδεσης στο Internet και απαιτεί ιδιαίτερη προσοχή στις λεπτομέρειες της εφαρμογής της. Το NAT χρησιμοποιεί διάφορες τεχνικές για την αντιμετώπιση αυτών των ειδικών περιπτώσεων και την επίδρασή του στην κίνηση στο δικτύου.

3.6.2 ICE

Το ICE (Interactive Connectivity Establishment) είναι μια τεχνική που έχει σκοπό να βρει τους διαθέσιμους τρόπους με τους οποίους δυο υπολογιστές μπορούν να συνδεθούν μεταξύ τους με όσο το δυνατόν πιο άμεσο τρόπο, με σκοπό να εγκαθιδρύσουν peer-to-peer επικοινωνία.

Η τεχνική ICE χρησιμοποιείται πολύ συχνά σε διαδραστικά μέσα, όπως για παράδειγμα, το Voice over Internet Protocol (VoIP), peer-to-peer επικοινωνία, βίντεο και instant messaging. Σε τέτοιες εφαρμογές, είναι επιθυμητό να αποφεύγεται η επικοινωνία μέσω ενός κεντρικού εξυπηρετητή, ο οποίος θα επιβραδύνει την επικοινωνία και είναι ακριβός. Έτσι, απαιτείται άμεση επικοινωνία μεταξύ των εφαρμογών πελάτη στο Διαδίκτυο, το οποίο όμως είναι πολύ δύσκολο λόγω των NATs, firewalls, και άλλων εμπόδιων δικτύου.

Αποτελείται από δύο υπο-πρωτόκολλα, το STUN (Session Traversal Utilities for NAT) και το TURN (Traversal Using Relays around NAT)

3.6.3 STUN

Το Session Traversal Utilities for NAT (STUN) είναι ένα τυποποιημένο σύνολο μεθόδων, συμπεριλαμβανομένου ενός πρωτοκόλλου δικτύου, για την διαχείριση της μετάφρασης διευθύνσεων δικτύου (NAT) σε εφαρμογές πραγματικού χρόνου φωνής, βίντεο, μηνυμάτων και άλλων τρόπων διαδραστικής επικοινωνίας.

Το STUN είναι ένα εργαλείο που χρησιμοποιείται από άλλα πρωτόκολλα, όπως το Interactive Connectivity Establishment (ICE), το Session Initiation Protocol (SIP) ή το WebRTC. Παρέχει ένα εργαλείο, χρησιμοποιούμενο από τον host για την ανακάλυψη μηχανισμού μετάφρασης διευθύνσεων δικτύου, και την χαρτογράφηση τους σε μία, συνήθως, δημόσια διεύθυνση IP και θύρας (port), τα οποία το πρωτόκολλο NAT έχει δεσμεύσει για τη χρήστη της εφαρμογής Datagram Protocol (UDP). Το πρωτόκολλο απαιτεί βοήθεια από έναν απομακρυσμένο server (STUN server) που βρίσκεται στην “απέναντι” (δημόσια) πλευρά του NAT.

Με το STUN ανοίγονται συνδέσεις προς ένα server στέλνοντας του UDP πακέτα και αυτός μας επιστρέφει σαν απάντηση πακέτα που περιλαμβάνουν την δημόσια IP μας και τις πόρτες από τις οποίες του στείλαμε το κάθε πακέτο. Αυτές οι πόρτες περιμένουμε να παραμείνουν ανοιχτές για νέα εισερχόμενα πακέτα και είναι αυτοί οι συνδυασμοί (IP,port) που στέλνουμε σε αυτούς που θέλουν να συνδεθούν μαζί μας. Η διαδικασία αυτή με τον STUN server συμβαίνει για παραπάνω από ένα πακέτα ώστε να συλλέξουμε όλες τις υποψήφιες IP και να δοκιμαστούν διάφοροι συνδυασμοί πριν αποκλείσουμε την δυνατότητα δημιουργίας σύνδεσης.

3.6.4 TURN

Το πρωτόκολλο Traversal Using Relays around NAT (TURN) χρησιμοποιείται βοηθητικά για τη διαχείριση της μετάφρασης διευθύνσεων δικτύου (NAT) ή των firewall για εφαρμογές πολυμέσων. Μπορεί να χρησιμοποιηθεί τόσο με το πρωτόκολλο Transmission Control Protocol (TCP) όσο και με το πρωτόκολλο User Datagram Protocol (UDP). Είναι χρήσιμο για τους πελάτες σε δίκτυα που χρησιμοποιούν NAT. Το πρωτόκολλο TURN υποστηρίζει τη σύνδεση ενός χρήστη πίσω από το NAT με ένα μοναδικό peer, όπως για παράδειγμα στην τηλεφωνία.

3.7 Kurento

Το Kurento είναι ένας WebRTC διακομιστής πολυμέσων (media server) και ένα σύνολο από client APIs τα οποία καθιστούν ευκολότερη την ανάπτυξη προχωρημένων εφαρμογών βίντεο για συσκευές που συνδέονται στο παγκόσμιο ιστό. Τα χαρακτηριστικά του περιλαμβάνουν ομαδική επικοινωνία, αλλαγή της κωδικοποίησης του βίντεο (transcoding), καταγραφή, μίξη, αναμετάδοση και δρομολόγηση οπτικοακουστικών ροών δεδομένων.

Παρέχει επίσης προχωρημένες δυνατότητες επεξεργασίας πολυμέσων όπως μηχανική όραση, επαυξημένη πραγματικότητα και ανάλυση ομιλίας. Η αρθρωτή αρχιτεκτονική του Kurento κάνει εύκολη την ενσωμάτωση αλγορίθμων επεξεργασίας πολυμέσων από τρίτους, όπως αναγνώριση ομιλίας, ανάλυση συναισθήματος, αναγνώριση προσώπου και άλλα, τα οποία μπορούν να χρησιμοποιηθούν διαφανώς από τους

προγραμματιστές εφαρμογών όπως και τα υπόλοιπα ενσωματωμένα χαρακτηριστικά του Kurento.

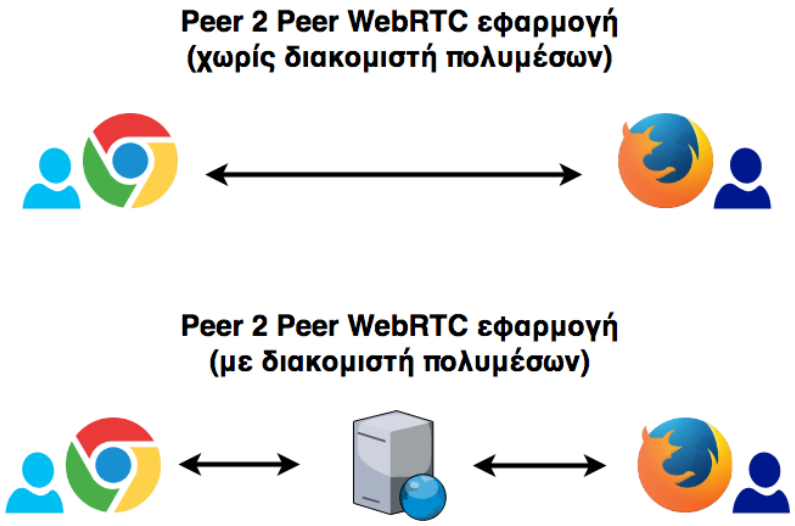
Ο πυρήνας του Kurento είναι ο Kurent Media Server (KMS), ο οποίος είναι υπεύθυνος για την μετάδοση των πολυμέσων, την επεξεργασία τους, το φόρτωμα τους και την καταγραφή τους. Υλοποιείται με τεχνολογίες χαμηλού επιπέδου που βασίζονται στο GStreamer για λόγους βελτιστοποίησης της κατανάλωσης πόρων. Το Kurento παρέχει τα παρακάτω χαρακτηριστικά:

- Δικτυακά πρωτόκολλα ροής συμπεριλαμβανομένου του HTTP, RTP και WebRTC
- Ομαδική επικοινωνία με λειτουργικότητα τόσο MCU (Multipoint Control Unit) όσο και SFU (Selective Forwarding Unit), υποστηρίζοντας δηλαδή και μίξη και δρομολόγηση πολυμέσων.
- Γενικότερη υποστήριξη για υπολογιστική όραση και φίλτρα επαυξημένης πραγματικότητας.
- Χώρο αποθήκευσης που υποστηρίζει διεργασίες εγγραφής για αρχεία WebM και MP4 και αναπαραγωγής για όλα τα είδη αρχείων που υποστηρίζονται από το GStreamer.
- Αυτόματη διακωδικοποιήση πολυμέσων ανάμεσα σε οποιοδήποτε από τους κωδικοποιητές (codecs) που υποστηρίζονται από το Gstreamer όπως είναι οι VP8, H.264, H.263, AMR, OPUS, Speex, G.711 και άλλοι.

Υπάρχουν διαθέσιμες βιβλιοθήκες του Kurento Client σε Java και Javascript για την διαχείριση του Kurento Media Server μέσα από την εκάστοτε εφαρμογή που έχει αναπτυχθεί αλλά μπορεί να χρησιμοποιηθεί και οποιαδήποτε άλλη γλώσσα προγραμματισμού επικοινωνώντας με τον Kurento Media Server σύμφωνα με το Kurento Protocol το οποίο βασίζεται σε JSON-RPC πάνω από Websockets.

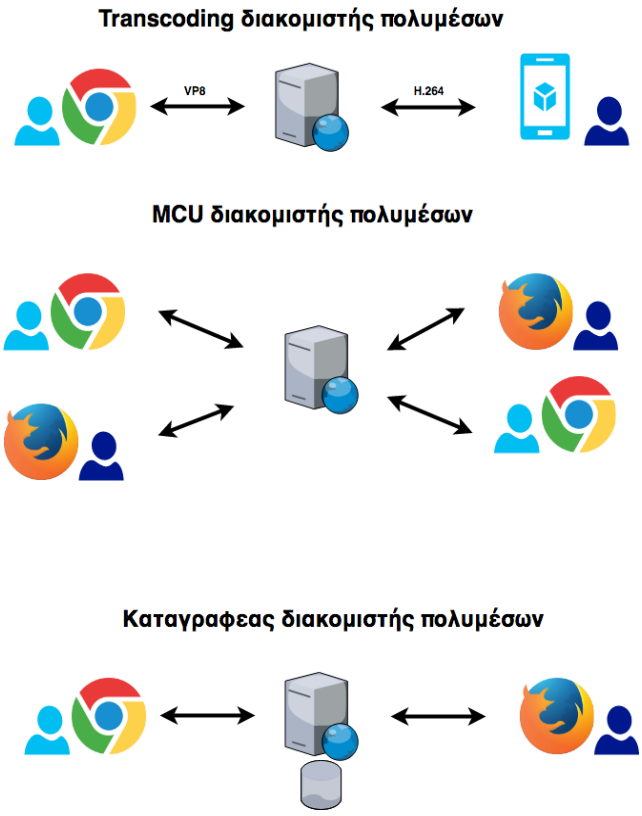
Το WebRTC όπως αναφέραμε νωρίτερα είναι μια τεχνολογία η οποία παρέχει την δυνατότητα στους browsers για επικοινωνία ζωντανού χρόνου μέσω των Javascript APIs της. Δημιουργήθηκε ως μια peer-to-peer τεχνολογία με την οποία οι browsers μπορούν να επικοινωνήσουν απευθείας μεταξύ τους χωρίς την παρέμβαση οποιασδήποτε άλλης υποδομής. Το μοντέλο αυτό είναι αρκετό για την δημιουργία απλών εφαρμογών αλλά επιπλέον λειτουργίες όπως είναι η ομαδική επικοινωνία, η καταγραφή της επικοινωνίας, η αναμετάδοση ή και η διακωδικοποίηση της ροής

δεδομένων, είναι δύσκολο να υλοποιηθούν με βάση αυτό. Για τον λόγο αυτό πολλές εφαρμογές απαιτούν την χρήση ενός media server.



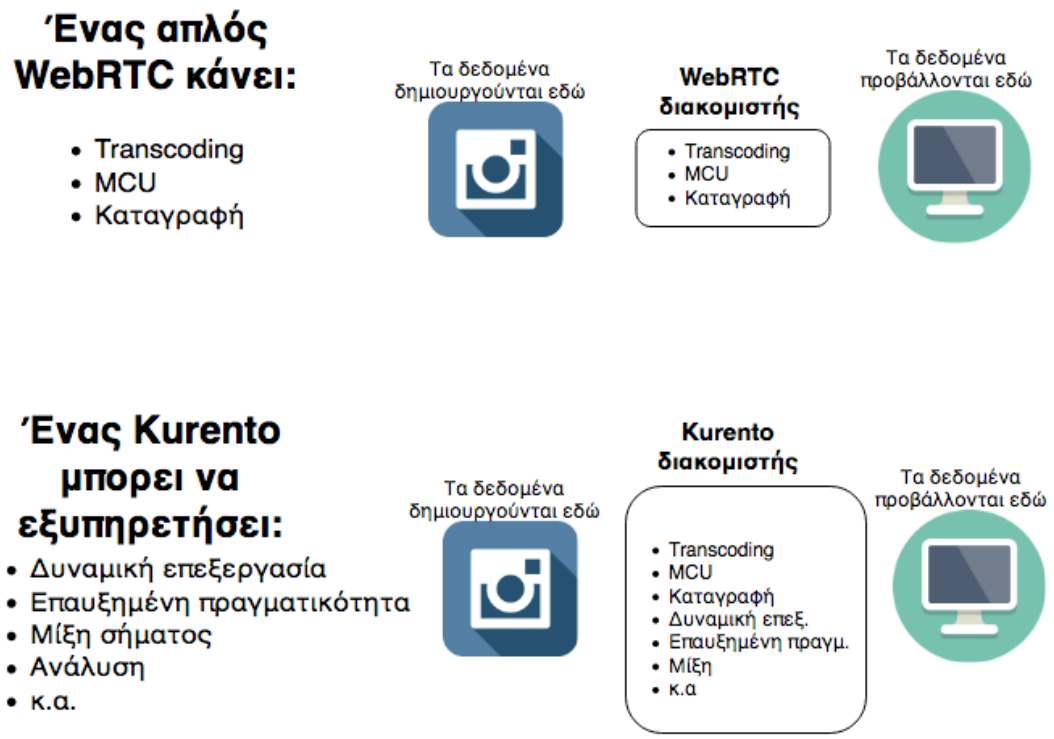
Εικόνα 11: WebRTC και Media Server

Ενοιολογικά, ένας WebRTC media server είναι ένας είδος ενδιάμεσου λογισμικού, με την λογική πως βρίσκεται ανάμεσα στους κόμβους που επικοινωνούν, από το οποίο περνάει η ροή πολυμέσων όταν φεύγει από την πηγή που την δημιούργησε και φτάνει στον προορισμό. Οι media servers είναι ικανοί να επεξεργαστούν ροές πολυμέσων και να προσφέρουν διάφορους τύπους στην έξοδο, να κατανείμουν την ροή πολυμέσων ενός peer σε πολλαπλούς δέκτες λειτουργώντας ως μια μονάδα πολλαπλής επικοινωνίας (Multi-Conference Unit, MCU), να κάνουν μίξη πολλαπλών εισερχομένων ροών σε μια μοναδική σύνθετη ροή, να μεταποιήσουν την κωδικοποίηση των πολυμέσων ώστε να την καταστήσει συμβατή με την συσκευή του αποδέκτη. Επίσης, μεταξύ άλλων, είναι ικανοί να καταγράψουν την συνομιλία αποθηκεύοντας τα πολυμέσα που ανταλλάσσονται με έναν μόνιμο τρόπο.



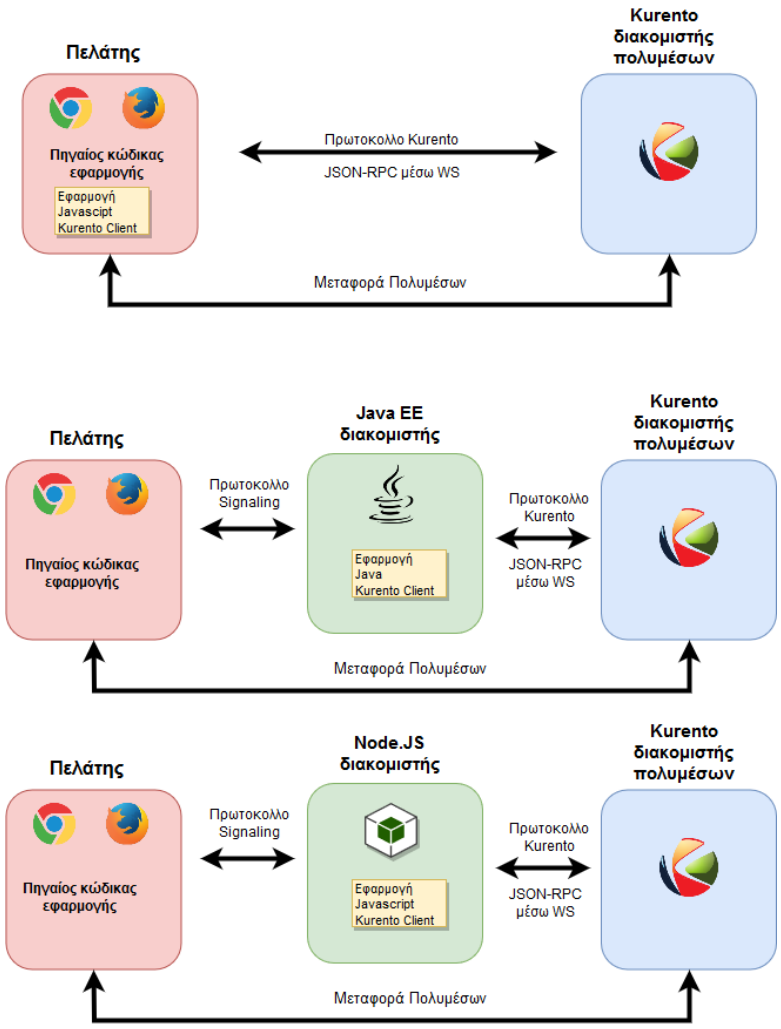
Εικόνα 12: Διάφοροι τύπου Media Servers

Ο Kurento Media Server, που βρίσκεται στον βάση της αρχιτεκτονικής του Kurento, βασίζεται σε συνδεόμενες (pluggable) μονάδες επεξεργασίας υπό την έννοια πως κάθε χαρακτηριστικό που προσφέρει είναι μια μονάδα η οποία συνδέεται πάνω του και μπορεί να ενεργοποιηθεί ή να απενεργοποιηθεί κατά βούληση. Επιπρόσθετα, οι προγραμματιστές μπορούν να δημιουργούν αδιαλείπτως πρόσθετες μονάδες που επεκτείνουν τον KMS με νέες λειτουργικότητες οι οποίες συνδέονται δυναμικά πάνω του.



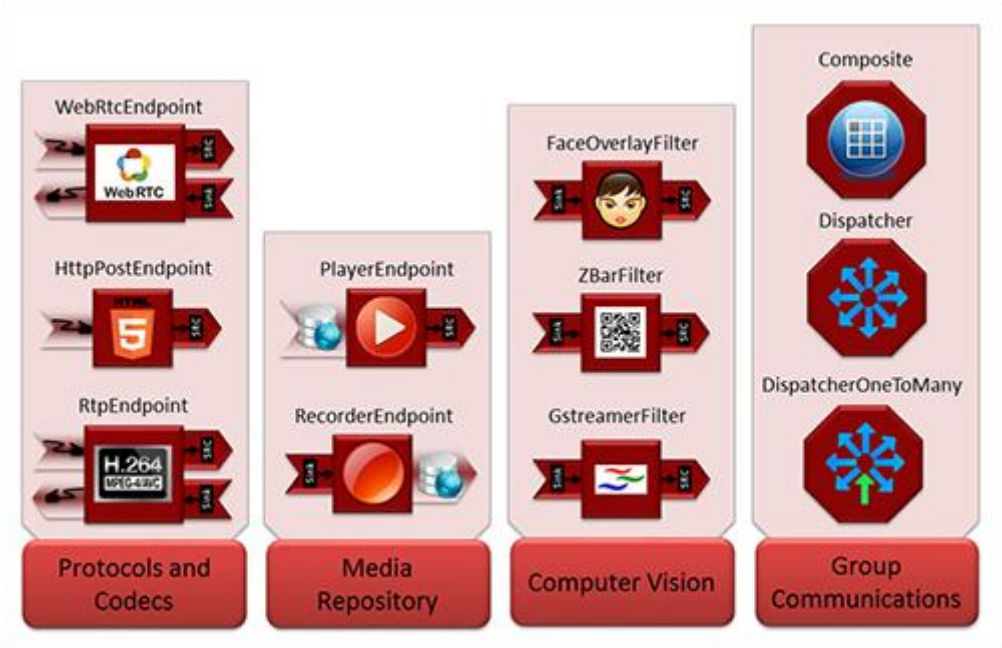
Εικόνα 13: Kurento Media Server

Οι δυνατότητες του KMS γίνονται διαθέσιμες στους προγραμματιστές μέσω του Kurento API. Η συγκεκριμένη διεπαφή υλοποιείται με την μορφή βιβλιοθηκών που ονομάζονται Kurento Clients. Το Kurento προσφέρει βιβλιοθήκες γραμμένες σε Javascript και Java αλλά υπάρχει η δυνατότητα χρήσης οποιασδήποτε άλλη γλώσσας προγραμματισμού ακολουθώντας απευθείας το πρωτόκολλο του Kurento. Οπότε υπάρχουν τρία σενάρια χρήσης της διεπαφής του Kurento όπως φαίνονται και στην εικόνα 14.



Εικόνα 14: Σενάρια χρήσης της διεπαφής του Kurento

Η διεπαφή του Kurento Client βασίζεται στην έννοια του στοιχείου πολυμέσων (Media Element). Ένα Media Element έχει σαφώς καθορισμένες δυνατότητες, για παράδειγμα το media element WebRtcEndpoint έχει την δυνατότητα να στέλνει και να δέχεται WebRTC ροές πολυμέσων, το media element RecorderEndpoint έχει την δυνατότητα να καταγράφει στο σύστημα αρχείων τις ροές πολυμέσων που δέχεται, το FaceOverlayFilter αναγνωρίζει τη θέση των προσώπων στα βίντεο που ανταλλάσσονται και προσθέτει πάνω τους μια συγκεκριμένη εικόνα. Το Kurento παρέχει μια πληθώρα τέτοιων στοιχείων μέσω των διεπαφών του.



Εικόνα 15: Media Element διαθέσιμα στην διεπαφή του Kurento Client[4]

3.8 MongoDB

Η MongoDB είναι μια open source βάση δεδομένων εγγραφών η οποία ανήκει στην κατηγορία των NoSQL βάσεων δεδομένων, αυτό σημαίνει πως τα δεδομένα δεν αποθηκεύονται σε πίνακες, όπως στις παραδοσιακές σχεσιακές βάσεις δεδομένων, αλλά στην περίπτωση την MongoDB αποθηκεύονται σε σχήματα (schemas) με μορφή παρόμοια με αυτή του προτύπου JSON (JavaScript Object Notation). Στην πραγματικότητα, η MongoDB είναι μία βάση δεδομένων που αποτελείται από μία συλλογή εγγράφων στα οποία αποθηκεύονται οι απαιτούμενες πληροφορίες.

Η μορφή των δεδομένων που αποθηκεύονται σε μία βάση δεδομένων MongoDB ποικίλει. Μπορεί να είναι είτε έγγραφα είτε κάποια δομή δεδομένων, ενώ είναι δυνατή η αλλαγή τους με την πάροδο του χρόνου. Τα δεδομένα που αποθηκεύονται αντιστοιχούν σε αντικείμενα στον κώδικα της εφαρμογής, κάνοντας τα ευέλικτα στην επεξεργασία. Η MongoDB προσφέρει τα ακόλουθα χαρακτηριστικά:

Ad hoc ερωτήματα

[4] https://doc-kurento.readthedocs.io/en/stable/user/features.html

Η MongoDB υποστηρίζει πεδία, ερωτήματα και regular expression για αναζήτηση δεδομένων. Τα ερωτήματα μπορούν να επιστρέφουν συγκεκριμένα πεδία των εγγράφων και μπορεί επίσης, να περιλαμβάνουν συναρτήσεις που ορίζονται από συναρτήσεις της JavaScript. Τα ερωτήματα μπορεί επίσης να ρυθμιστούν να επιστρέφουν ένα τυχαίο δείγμα αποτελεσμάτων ενός δεδομένου μεγέθους.

Indexing

Τα πεδία σε ένα έγγραφο MongoDB μπορούν να πάρουν την μορφή ευρετηρίου χρησιμοποιώντας πρωτογενείς και δευτερεύοντες δείκτες. Η κατηγοριοποίηση των δεδομένων σε πραγματικό χρόνο, δίνει τη δυνατότητα και τα εργαλεία για πρόσβαση και ανάλυση των δεδομένων.

Replication

Το MongoDB παρέχει υψηλή διαθεσιμότητα μέσω αντιγράφων (replica sets). Ένα replica set αποτελείται από δύο ή περισσότερα αντίγραφα δεδομένων. Κάθε replica set μπορεί να έχει το ρόλο πρωτεύοντος ή δευτερεύοντος αντιγράφου κάθε στιγμή. Η εγγραφή και η ανάγνωση δεδομένων γίνονται στο πρωτεύων αντίγραφο. Τα δευτερεύοντα αντίγραφα διατηρούν ένα αντίγραφο των πρωτευόντων δεδομένων με μια ενσωματωμένη τεχνική αντιγραφής. Όταν αποτύχει ένα πρωτεύων αντίγραφο, διεξάγει αυτόματα μια εκλογική διαδικασία στο σύνολο των replica sets για να προσδιοριστεί ποιο δευτερεύον replica set πρέπει να γίνει το κύριο. Τα δευτερεύοντα τμήματα μπορούν προαιρετικά να προσφέρουν λειτουργίες ανάγνωσης, αλλά αυτά τα δεδομένα δεν είναι απαραίτητα συνεπή.

Load balancing

Το MongoDB προσφέρει οριζόντια κλιμάκωση με τη χρήση sharding. Ο χρήστης επιλέγει ένα sharded κλειδί, το οποίο καθορίζει πως τα δεδομένα μιας συλλογής θα διανεμηθούν. Τα δεδομένα χωρίζονται σε εύρη τιμών (με βάση το sharded κλειδί) και κατανέμονται μεταξύ πολλαπλών shards. Ένα shard είναι master με έναν ή περισσότερους slaves. Εναλλακτικά, το shard κλειδί μπορεί να επιτρέψει μια ομοιόμορφη κατανομή δεδομένων.

Το MongoDB μπορεί να τρέχει παράλληλα σε πολλαπλούς servers, εξισορροπώντας το φορτίο ή αντιγράφοντας τα δεδομένα για να διατηρήσει το σύστημα σε λειτουργία σε περίπτωση αποτυχίας υλικού.

Αποθήκευση αρχείων (File Storage)

Η MongoDB μπορεί να χρησιμοποιηθεί σαν ένα σύστημα αρχείων με λειτουργίες εξισορρόπησης φορτίου και αντιγραφής δεδομένων σε πολλαπλά μηχανήματα για την αποθήκευση αρχείων. Η λειτουργία αυτή ονομάζεται grid file system. Η MongoDB διαθέτει λειτουργίες επεξεργασίας των αρχείων και του περιεχομένου τους που μπορούν να χρησιμοποιηθούν από προγραμματιστές.

Aggregation

Το MapReduce μπορεί να χρησιμοποιηθεί για την επεξεργασία δεδομένων σε ομάδες και λειτουργίες συγκέντρωσης (aggregation). Η λειτουργία aggregation δίνει την δυνατότητα στους χρήστες να πάρουν αποτελέσματα παρόμοια με αυτά που προκύπτουν από το SQL GROUP BY. Επιπλέον, η λειτουργία aggregation δίνει την δυνατότητα σχηματισμού pipeline αντίστοιχο του Unix. Τέλος, η aggregation προσφέρει τη λειτουργία $lookup για τη συνένωση εγγράφων από πολλαπλά έγγραφα.

Χρήση JavaScript στην πλευρά του server

Η JavaScript μπορεί να χρησιμοποιηθεί σε ερωτήματα, aggregation συναρτήσεις που αποστέλλονται απευθείας στη βάση δεδομένων προς εκτέλεση.

Capped collections

Η MongoDB υποστηρίζει συλλογές σταθερού μεγέθους που ονομάζονται capped collections. Αυτός ο τύπος συλλογής επιτρέπει την εισαγωγή δεδομένων με τη σειρά και φτάσει σε ένα καθορισμένο μέγεθος, συμπεριφέρεται σαν circular queue.

Με βάση όλα τα παραπάνω χαρακτηριστικά, η MongoDB είναι μια κατανεμημένη βάση δεδομένων που είναι εύκολη στη χρήση.

3.9 NodeJS

Το Node.js είναι μια πλατφόρμα ανάπτυξης λογισμικού για την εκτέλεση εφαρμογών υλοποιημένες σε Javascript. Δημιουργήθηκε το 2009 από τον Ryan Dahl με σκοπό την ανάπτυξη διαδικτυακών εφαρμογών σε Javascript στην πλευρά του εξυπηρετητή (server-side Javascript). Η Javascript μέχρι τότε ήταν μια γλώσσα που χρησιμοποιούταν κυρίως για εφαρμογές στην πλευρά του πελάτη (client-side Javascript), όμως το Node.js μετέφερε αυτή την δυνατότητα και στην πλευρά του εξυπηρετητή με την χρήση της V8, μιας Javascript μηχανής που αναπτύχθηκε από την Google για την αύξηση της απόδοσης στην εκτέλεση της Javascript για τις ανάγκες του Chrome και την οποία στην συνέχεια διένειμε σαν ξεχωριστή οντότητα. Η V8 καταφέρνει να αυξήσει την απόδοση κάνοντας απευθείας μεταγλώττιση του πηγαίου κώδικα της Javascript σε γλώσσα μηχανής πριν την εκτέλεση του, αντί να εκτελεί bytecode ή να χρησιμοποιεί διερμηνευτή (interpreter).

Η φιλοσοφία πίσω από το Node.js ήταν η δημιουργία μιας πλατφόρμας η οποία είναι οδηγούμενη από τα γεγονότα (event-driven) και στην οποία κατά την διάρκεια εκτέλεσης του κώδικα όταν υπάρχει η ανάγκη για Είσοδο / Έξοδο δεδομένων δεν χρειάζεται η εφαρμογή να περιμένει το τέλος αυτής της διαδικασίας για να συνεχίσει στις επόμενες εντολές (non-blocking I/O) αντιθέτως μπορεί να εκτελεστεί ασύγχρονα (asychronous I/O) και όταν αυτή ολοκληρωθεί να εκτελεστεί και η αντίστοιχη συνάρτηση που έχει ανατεθεί στο αποτέλεσμα της Εισόδου / Εξόδου. Με αυτή την σχεδιαστική επιλογή το Node.js στοχεύει στην βελτιστοποίηση της απόδοσης και της κλιμακοσιμότητας των διαδικτυακών εφαρμογών, οι οποίες έχουν πολλές διεργασίες εισόδου/εξόδου.

Ο οδηγούμενος από γεγονότα προγραμματισμός είναι ένα προγραμματιστικό μοντέλο στο οποίο η ροή εκτέλεσης ορίζεται από τα γεγονότα. Τα γεγονότα χειρίζονται από τους χειριστές γεγονότων (event handlers) ή από τις επιστρεφόμενες κλήσεις γεγονότων (event callbacks). Μια επιστρεφόμενη κλήση γεγονότος είναι μια συνάρτηση η οποία καλείται όταν συμβεί κάποιο σημαντικό γεγονός όπως είναι μια λειτουργία Εισόδου / Εξόδου. Σε αυτό το είδος προγραμματισμού οι λειτουργίες Εισόδου / Εξόδου δεν επιστρέφουν μια τιμή αλλά μια συνάρτηση (callback function).

4. Υλοποίηση

4.1 Αρχιτεκτονική

Η αρχιτεκτονική του Kurento, όπως και τον περισσοτέρων τεχνολογιών πολυμεσικής επικοινωνίας, δομείται χρησιμοποιώντας δύο επίπεδα ως αφαιρετικές έννοιες που συνοψίζουν τις βασικές λειτουργίες στα διαδραστικά συστήματα επικοινωνίας:

Επίπεδο Σηματοδοσίας (Signaling Plane)

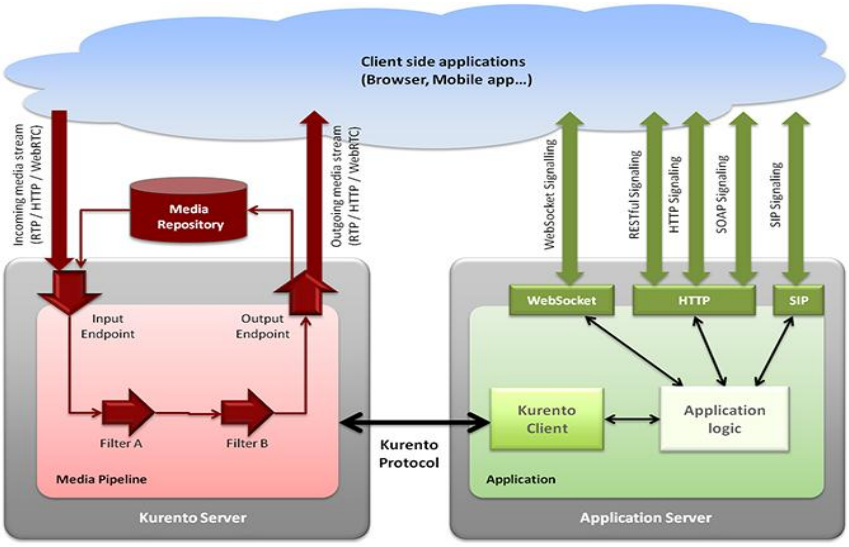
Τα επιμέρους τμήματα του συστήματος τα οποία είναι αρμόδια για την διαχείριση της επικοινωνίας, δηλαδή οι μονάδες αυτές που παρέχουν λειτουργίες σχετικά με την διαπραγμάτευση των πολυμέσων, την παραμετροποίηση του QoS, την εγκαθίδρυση της κλήσης, την εγγραφή των χρηστών, την παρουσία των χρηστών

Επίπεδο Πολυμέσων (Media Plane)

Λειτουργικότητες όπως η μεταφορά των πολυμέσων, η κωδικοποίηση / αποκωδικοποίηση των πολυμέσων, η επεξεργασία των πολυμέσων απαρτίζουν το Επίπεδο Πολυμέσων, το οποίο αναλαμβάνει την διαχείριση των πολυμέσων.

Ο διαχωρισμός αυτός προέρχεται από τον τηλεφωνία στην οποία έχουμε διάκριση ανάμεσα στην διαχείριση της φωνής και στην διαχείριση των μετα-δεδομένων όπως είναι η χρέωση και ο τόνος που ακούγεται όταν καλούμε.

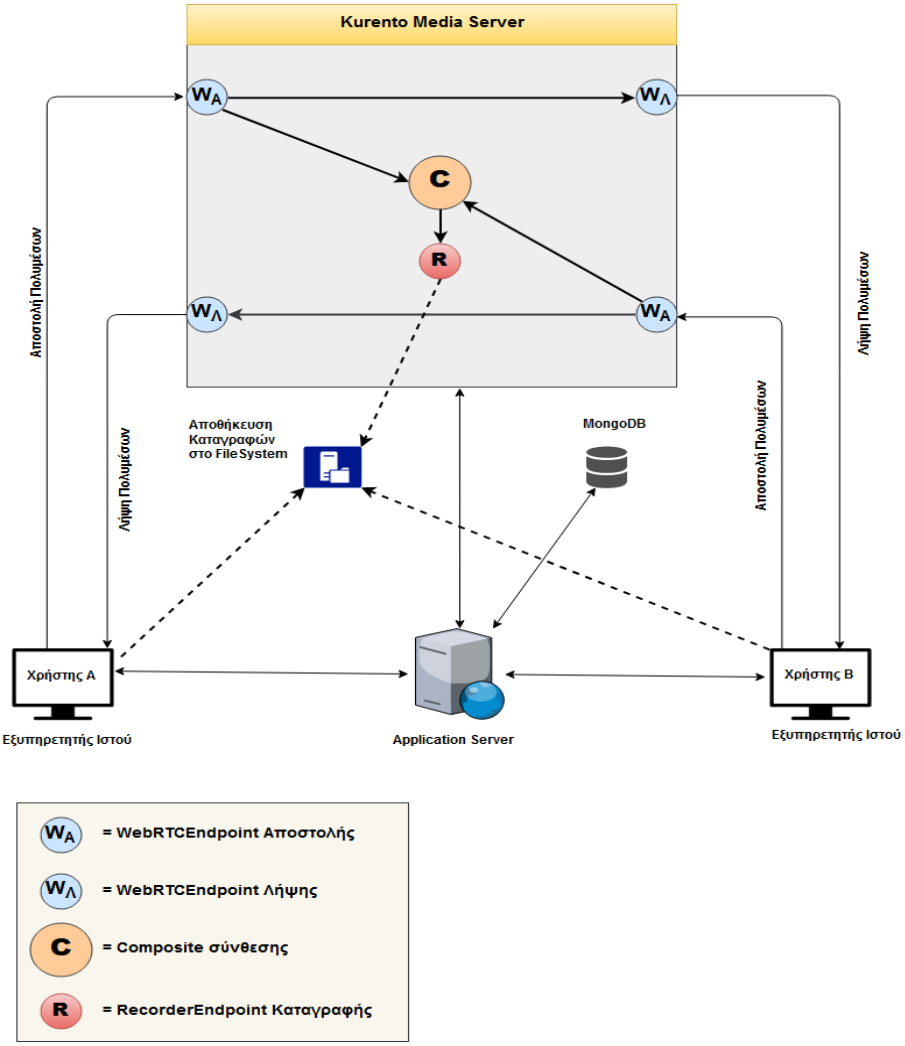
Στην ακόλουθη εικόνα βλέπουμε μια εννοιολογική αναπαράσταση της αρχιτεκτονικής του Kurento.



Εικόνα 16: Εννοιολογική αναπαράσταση αρχιτεκτονικής του Kurento[5]

Στην εφαρμογή μας το επίπεδο σηματοδοσίας αναλαμβάνει ο application server και το επίπεδο πολυμέσων αναλαμβάνει ο Kurento Media Server. Στην εικόνα 17 παρουσιάζεται η αρχιτεκτονική του συστήματος για την περίπτωση της επικοινωνίας μεταξύ δυο χρηστών. Θα αναλύσουμε παρακάτω τα επιμέρους στοιχεία του συστήματος.

[5] http://doc-kurento.readthedocs.io/en/stable/user/writing_applications.html#global-architecture

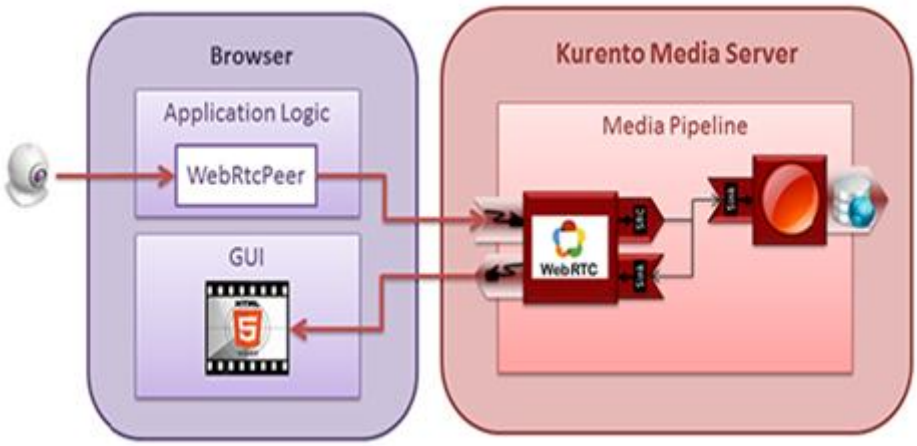


Εικόνα 17:Ενσωμάτωση όλων των στοιχείων με τον Kurento Media Server και οι διαύλοι επικοινωνίας μεταξύ τους

4.1.1 Πλευρά Εξυπηρετητή

4.1.1.1 Kurento Media Server

Στη βάση του επιπέδου πολυμέσων βρίσκεται ο Kurento Media Server ο οποίος λειτουργεί με την λογική της σωλήνωσης (pipeline). Για κάθε επικοινωνία δημιουργείται μια σωλήνωση η οποία είναι μια ακολουθία διαφορετικών στοιχείων που συνδέονται για να σχηματίσουν την διαδρομή που θα ακολουθήσουν τα δεδομένα πολυμέσων.



Εικόνα 18: Kurento Media Server και η λογική της σωλήνωσης (pipeline)[6]

Ο Kurento Media Server είναι αυτός που προσφέρει τα εργαλεία για την υλοποίηση του συστήματος μας. Τα βασικά εργαλεία που χρησιμοποιούμε από την εργαλειοθήκη που μας δίνει είναι:

WebRtcEndpoint*:*

Εικόνα 19: WebRtcEndpoint[7]

Το WebRtcEndpoint είναι ένα αμφίδρομο στοιχείο πολυμέσων (media element) το οποίο έχει την δυνατότητα να στέλνει και να λαμβάνει WebRTC ροές δεδομένων. Υλοποιεί όλα τα απαραίτητα WebRTC πρωτόκολλα συμπεριλαμβανομένου του ICE, SRTP, DLS και μπορεί να διαχειριστεί κωδικοποιήσεις βίντεο όπως είναι οι VP8 και OPUS. Το WebRtcEndpoint έχει δοκιμαστεί με τις υλοποιήσεις WebRTC που έχουν εφαρμοστεί στους φυλλομετρητές Chrome και Firefox και είναι πλήρως συμβατό μαζί του. Επομένως, όσον αφορά τον φυλλομετρητή, το WebRtcEndpoint συμπεριφέρεται σαν ένας WebRTC peer με τον οποίο συνδέεται ο χρήστης.

[6] http://builds.kurento.org/dev/master/latest/docs/_images/media-pipeline-sample.png
[7] https://doc-kurento.readthedocs.io/en/stable/features/kurento_api.html

Από την μεριά του media server, το WebRtcEndpoint παρέχει ένα στρώμα πηγής πολυμέσων (source media pad), στο οποίο τα WebRTC πολυμέσα που έρχονται από το δίκτυο γίνονται διαθέσιμα σε άλλα στοιχεία πολυμέσων. Παρέχει επίσης κι ένα στρώμα "καταβόθρας" πολυμέσων (sink media pad) με σκοπό η ροή πολυμέσων με την οποία θα το τροφοδοτήσουν άλλα στοιχεία πολυμέσων να αποτελέσει την ροή που θα λάβει ο απομακρυσμένος peer του δικτύου που έχει συνδεθεί με το WebRtcEndpoint.

Composite:

Εικόνα 20: Composite[8]

Το Composite αποτελείται από ένα πλήθος θυρών στις οποίες συνδέονται τα στοιχεία πολυμέσων δρομολογώντας την κίνηση τους σε αυτό τον κόμβο. Μέσα στον κόμβο γίνεται μίξη του ήχου όλων των συνδεδεμένων ροών και δημιουργείται ένα πλέγμα με τα βίντεο των ροών. Με αυτό τον τρόπο αποκτάμε πρόσβαση σε μια ενιαία ροή την οποία μπορούμε πλέον να αποθηκεύσουμε τροφοδοτώντας την σε ένα στοιχείο τύπου RecorderEndpoint.

RecorderEndpoint:

Εικόνα 21: RecorderEndpoint[9]

Το RecorderEndpoint είναι ένα στοιχείο πολυμέσων το οποίο δίνει την δυνατότητα αποθήκευσης των πολυμέσων που τροφοδοτούνται σε αυτό. Άλλα στοιχεία

[8] https://doc-kurento.readthedocs.io/en/stable/features/kurento_api.html
[9] https://doc-kurento.readthedocs.io/en/stable/features/kurento_api.html

συνδέονται δηλαδή πάνω του και το τροφοδοτούν με τα πολυμέσα τους τα οποία στη συνέχεια αποθηκεύονται με μόνιμο τρόπο.

Στην περίπτωση μας λοιπόν για την δημιουργία της κλήσης ακολουθούνται τα παρακάτω βήματα:

1. Δημιουργία MediaPipeline για την κλήση
2. Δημιουργία στοιχείου Composite
3. Δημιουργία θύρας στο composite για το RecorderEndpoint
4. Δημιουργία στοιχείου WebrtcEndpoint (outgoing από την μεριά του χρήστη) το οποίο δημιουργεί p2p σύνδεση με τον χρήστη και λαμβάνει τα δεδομένα πολυμέσων του
5. Δημιουργία θύρας στο composite για να προωθηθούν σε αυτό τα δεδομένα του χρήστη
6. Δημιουργία στοιχείων WebrtEndpoint (incoming από την μεριά του χρήστη), ίσα στον αριθμό με το πλήθος των υπολοίπων συμμετεχόντων στην συνομιλία, τα οποία δημιουργούν p2p σύνδεση με τον χρήστη και του στέλνουν τα δεδομένα των άλλων χρηστών
7. Προώθηση των δεδομένων του κάθε χρήστη στους υπολοίπους συνδέοντας το στοιχείο που δημιουργήθηκε στο βήμα 4 με αυτά που δημιουργήθηκαν στο βήμα 6.

Τα βήματα 1 έως 3 εκτελούνται μια φορά στην εκκίνηση της συνομιλίας ενώ τα βήματα 4 έως 7 εκτελούνται μια φορά για κάθε χρήστη που συμμετέχει στην συνομιλία.

Στην περίπτωση της καταγραφής υπάρχουν τα εξής επιπλέον βήματα:

8. Δημιουργία RecorderEndpoint
9. Σύνδεση της θύρας του composite που δημιουργήθηκε στο βήμα 2 με το στοιχείο RecorderEndpoint που δημιουργήθηκε στο βήμα 8.

Στη υλοποίηση μας, ο Kurento media server δεν υποστηρίζει τη δυνατότητα μεταφοράς μεταδεδομένων. Τα μεταδεδομένα που κρίνονται απαραίτητα για την πραγματοποίηση της καταγραφής διαχειρίζονται (μεταφέρονται, διαβάζονται, αποθηκεύονται) από το javascript κομμάτι της εφαρμογής μας. Ένα τέτοιο

παράδειγμα αποτελεί η περίπτωση των timestamp της καταγραφής της συνδιάσκεψης, όπως παρουσιάζονται παρακάτω, στο αντίστοιχο κομμάτι του πηγαίου κώδικα.

Καταγραφή timestamp έναρξης συνδιάσκεψης

```
room.pcall = new CallRecord
	({
		recordUri:"https://83.212.124.132:8081/records/"+recordFilename,
		startTime: Date.now(),
		participants: tmpPart
	})
```

Καταγραφή timestamp λήξης συνδιάσκεψης

```
room.pcall.endTime = Date.now();
```

4.1.1.2. Signaling

Το Signaling είναι η διαδικασία με την οποία συντονίζεται η επικοινωνία. Για να εγκαθιδρυθεί μια κλήση χρησιμοποιώντας το WebRTC οι συμμετέχοντες πρέπει να ανταλλάξουν πληροφορίες μεταξύ τους. Αυτές οι πληροφορίες περιλαμβάνουν:

- Μηνύματα ελέγχου της συνεδρίας
- Μεταδεδομένα πολυμέσων όπως η κωδικοποίηση του ήχου και της εικόνας, το εύρος της σύνδεσης, οι τύποι των πολυμέσων.
- Δεδομένα σχετικά με την εγκαθίδρυση ασφαλούς σύνδεσης
- Δικτυακά δεδομένα, όπως η IP διεύθυνση και η θύρα που θα χρησιμοποιηθεί

Η εφαρμογή αναπτύχθηκε σε Javascript και εκτελείται με την βοήθεια του Node.js. Έχει το ρόλο του διαχειριστή της επικοινωνίας και της σύνδεσης μεταξύ των επιμέρους οντοτήτων που απαρτίζουν την εφαρμογή. Η δομή στις εφαρμογές που αναπτύσσονται στο Node.js περιλαμβάνει τα αρχεία του κώδικα μας και ένα αρχείο package.json που δηλώνει τα επιμέρους πακέτα κώδικά (npm packages) τα οποία χρησιμοποιεί η εφαρμογή μας. Στην περίπτωση μας αυτά τα επιπλέον πακέτα είναι τα:

- express: web framework του node.js το οποίο χρησιμοποιείται για να σερβίρει ιστοσελίδες και τα αρχεία της στους πελάτες (clients)

- ws: WebSocket απαραίτητο για τον γρήγορο έλεγχο της εφαρμογής πελάτη – εξυπηρέτη.
- Kurento-client: Απαραίτητο για την επικοινωνία με τον Kurento media server μεσω RPCs
- Socket.io: Υλοποίηση Websocket server και client
- Bower: Πρόκειται για ένα εργαλείο διαχείρισης πακέτων για front-end βιβλιοθήκες, όπως τα jQuery, Bootstrap και ούτω καθεξής.
- Mongoose: Εργαλείο μοντελοποίησης αντικειμένων της βάσης MongoDB σχεδιασμένο να δουλεύει σε ασύγχρονο περιβάλλον

Ο application server είναι γραμμένος σε Javascript και εκτελείται με την βοήθεια του Node.js. Ο application server είναι ο αρμόδιος για να:

- διαχειριστεί τους χρήστες που έχουν συνδεθεί στην εφαρμογή διατηρώντας ένα διαρκή δίαυλο επικοινωνίας μαζί τους
- διαχειρίζεται τον KMS δίνοντας του τις κατάλληλες εντολές για την δημιουργία των κλήσεων
- επικοινωνεί με την βάση δεδομένων MongoDB για την αποθήκευση των δεδομένων της συνομιλίας

Για την διαχείριση των χρηστών χρησιμοποιούμε δυο modules που βοηθάνε στην οργάνωση της σύνδεσης μεταξύ των χρηστών και του server. Αυτά είναι:

User-session.js

Μια δομή στην οποία αποθηκεύει ο server προσωρινά τα στοιχεία των συνδεδεμένων χρηστών που είναι μοναδικά για αυτούς. Τα στοιχεία αυτά σχετίζονται με τη σύνδεση των χρηστών με το διακομιστή και αφορά τα endpoints που του αντιστοιχούν στον Kurento Media Server. Επιπλέον, περιέχει συναρτήσεις που βοηθούν στην ανταλλαγή μηνυμάτων μεταξύ του χρήστη και του διακομιστή.

```
function UserSession(id, socket) {
   this.id = id;
   this.socket = socket;
   this.outgoingMedia = null;
   this.hubPort = null;
   this.incomingMedia = {};
   this.iceCandidateQueue = {};
}
```

user-registry.js

Μια δομή η οποία λειτουργεί σαν κατάλογος για τα στοιχεία τύπου user-session. Περιέχει, επίσης, συναρτήσεις για τη διαχείριση των στοιχείων αυτών.

server.js

Κάθε χρήστης που ανοίγει την σελίδα της εφαρμογής δημιουργεί ένα κανάλι επικοινωνίας με τον application server με την χρήση του socket.io. Μέσω του καναλιού περιμένουμε μηνύματα από τον χρήστη τα οποία εκκινούν αντίστοιχες συναρτήσεις στον server για την υλοποίηση διάφορων λειτουργιών. Οι λειτουργίες αυτές σχετίζονται με την είσοδο του χρήστη, την εκκίνηση της επικοινωνίας και άλλα, τα οποία θα αναλύσουμε εκτενώς στο επόμενο κεφάλαιο.

4.1.1.3. MongoDB

Για τη διατήρηση των στοιχείων των χρηστών που λαμβάνουν μέρος στην επικοινωνία, διατηρείται μία βάση δεδομένων. Αυτή η βάση δεδομένων είναι μία MongoDB.

Για να είναι δυνατή η διαχείριση και η πρόσβαση της MongoDB, χρησιμοποιείται ένα object τύπου Mongoose. Το Mongoose που χρησιμοποιείται λειτουργεί σαν το front end της βάσης δεδομένων.

Ο συνδυασμός της χρήσης Mongoose με τη MongoDB είναι διαδεδομένος σε συστήματα επικοινωνίας όπως το WebRTC, καθώς τόσο ο τρόπος αποθήκευσης των δεδομένων όσο και τα ερωτήματα σε αυτά μοιάζουν αρκετά με JSON.

Σύνδεση στη Βάση Δεδομένων

Για να είναι δυνατή η επικοινωνία του server με τη βάση δεδομένων, απαιτείται η σύνδεση του Mongoose με τη βάση δεδομένων MongoDB. Αυτό πραγματοποιείται με τις ακόλουθες γραμμές κώδικα, οι οποίες βρίσκονται στο αρχείο server.js.

```
var mongoose = require('mongoose');
…
var settings = {
    …
    MONGODBURL: "mongodb://localhost:27017"
};
…
mongoose.connect(settings.MONGODBURL,function (error){
    if (error){
        console.log('ERROR connecting to: '+settings.MONGODBURL+'. '+error);
    }
    else{
        console.log('Succeded connecting to: '+settings.MONGODBURL);
    }
});
```

Αρχικά εισάγουμε το Mongoose module στον κώδικά μας μέσω της require(). Στη συνέχεια δηλώνουμε την διεύθυνση URL στην οποία είναι αποθηκευμένη η mongoDB. Στη συγκεκριμένη περίπτωση, η βάση δεδομένων είναι αποθηκευμένη τοπικά στο μηχάνημά μας. Τέλος, συνδεόμαστε στη βάση δεδομένων μέσω της mongoose.connect(). Για να είμαστε βέβαιοι σχετικά με την έκβαση της σύνδεσης μας, παράγεται ένα μήνυμα τόσο στην περίπτωση αποτυχίας όπου εμφανίζεται και το αντίστοιχο λάθος που συνέβη, όσο και στην περίπτωση επιτυχίας.

Δημιουργία μοντέλων

Για να είναι δυνατή η υλοποίηση της βάσης δεδομένων μας, χρησιμοποιούμε μοντέλα (models), τα οποία ορίζονται χρησιμοποιώντας διεπαφές σχήματος (scheme interface). Μέσω ενός σχήματος είναι δυνατόν να ορίσουμε τα πεδία που θα αποθηκεύονται σε κάθε έγγραφο μαζί με τις απαιτήσεις επικύρωσής τους και τις προεπιλεγμένες τιμές, που θα έχουν. Έτσι ορίζεται το σχήμα των εγγράφων που θα χρησιμοποιηθούν.

Στη συνέχεια, τα σχήματα συνδέονται με τα αντίστοιχα μοντέλα, μέσω της μεθόδου mongoose.model(). Κάθε μοντέλο, που υπάρχει στη βάση δεδομένων μας, μπορεί να χρησιμοποιηθεί για όλες τις λειτουργίες που παρέχονται στα αντικείμενα του συγκεκριμένου τύπου. Έτσι, ένα μοντέλο, μπορεί να χρησιμοποιηθεί για αναζήτηση (find), δημιουργία (create), ενημέρωση (update) και διαγραφή (delete) των αντικειμένων του.

Στο ακόλουθο κομμάτι κώδικα ορίζουμε το σχήμα που πρόκειται να χρησιμοποιηθεί. Το σχήμα αυτό ορίζει τα τέσσερα διαφορετικά πεδία: recordUri το οποίο είναι String και αποθηκεύει ένα μοναδικό αναγνωριστικό για κάθε επικοινωνία, την χρονική στιγμή έναρξης και λήξης της επικοινωνίας με τα πεδία startTime και endTime αντίστοιχα, και τέλος μία λίστα με τους συμμετέχοντες στην επικοινωνία αυτή στο πεδίο participants.

```
var callSchema = new mongoose.Schema({

    recordUri: String,

    startTime: Date,

    endTime: Date,

    participants:{type:[String],index:true}

});
```

Στη συνέχεια, για να είναι αξιοποιήσιμο το παραπάνω σχήμα, δημιουργούμε ένα μοντέλο, μέσω της συνάρτησης model(). Το πρώτο όρισμα της συνάρτησης είναι ένα μοναδικό όνομα το οποίο είναι και το όνομα της συλλογής που θα δημιουργηθεί από το μοντέλο αυτό. Το δεύτερο όρισμα ορίζει το σχήμα που θα χρησιμοποιηθεί για τη δημιουργία του μοντέλου. Έτσι, από το ακόλουθο κομμάτι κώδικα, τη συλλογή θα ονομαστεί CallRecord και θα βασίζεται στο προηγούμενο σχήμα.

```
var CallRecord = mongoose.model('CallRecord',callSchema);
```

4.1.2 Client Side

4.1.2.1.WebRTC API

Τα WebRTC APIs βασίζονται στο WebRTC 1.0 του W3C και επιτρέπουν επικοινωνία σε πραγματικό χρόνο μεταξύ των φυλλομετρητών και όχι μόνο. Αυτή τη στιγμή, το WebRTC χρησιμοποιείται σε διάφορες εφαρμογές όπως το WhatsApp, το Facebook Messenger, το appear.in και σε πλατφόρμες όπως το TokBox.

Ωστόσο, για να είναι αυτό εφικτό έχουν αναπτυχθεί τρία διαφορετικά APIs. Τα διαθέσιμα API σχετίζονται είναι τα MediaStream, RTCPeerConnection και RTCDataChannel. Το MediaStream έχει τη δυνατότητα πρόσβασης σε ροές δεδομένων, οι οποίες προέρχονται από την κάμερα και το μικρόφωνο του χρήστη. Το RTCPeerConnection υποστηρίζει κλήσεις ήχου ή βίντεο, με εγκαταστάσεις κρυπτογράφησης και δυνατότητα διαχείρισης του εύρου ζώνης. Το RTCDataChannel προσφέρει επικοινωνία μεταξύ γενικών δεδομένων από ομότιμους χρήστες. Στη συνέχεια, θα δούμε αναλυτικά τα APIs αυτά.

Media Stream API

Το MediaStream API χρησιμοποιείται για συγχρονισμένες ροές μέσων. Η διεπαφή MediaStreamTrack ορίζεται σαν μια ροή δεδομένων ήχου ή βίντεο και γενικότερα

μπορεί να επεκταθεί ώστε να αντιπροσωπεύει μια ροή πολυμέσων. Για παράδειγμα, το MediaStream API χρησιμοποιείται για μία ροή δεδομένων που λαμβάνεται από την είσοδο μίας κάμερας ή του μικροφώνου και έχει συγχρονισμένα κομμάτια βίντεο και ήχου. Τα πολυμέσα αυτά μπορεί είτε να προέρχονται είτε να αποστέλλονται σε έναν απομακρυσμένο συνομιλητή και όχι απαραίτητα στην τοπική κάμερα ενός χρήστη.

Κάθε MediaStream έχει μια είσοδο, η οποία συνήθως είναι ένα MediaStream που παράγεται από το navigator.getUserMedia(), και μια έξοδος, η οποία είτε μεταβιβάζεται σε ένα στοιχείο βίντεο ή μία RTCPeerConnection.

Η μέθοδος getUserMedia () παίρνει τρεις παραμέτρους:

- Ένα αντικείμενο contstraint. Το αντικείμενο contstraint έχει εφαρμογή στους Chrome, Firefox και Opera και χρησιμοποιείται για τον ορισμό διάφορων παραμέτρων όπως η ανάλυση βίντεο που θα χρησιμοποιηθεί κατά τη διάρκεια της κλήσης.
- Ένα επιτυχημένο callback. Το επιτυχημένο callback, χρησιμοποιείται όταν μια κλήση είναι επιτυχής και χρησιμοποιείται για να περάσει το MediaStream.
- Ένα αποτυχημένο callback. Το αποτυχημένο callback καλείται για να σταλθεί ένα αντικείμενο σφάλματος όταν η κλήση δεν είναι επιτυχής.

Κάθε MediaStream έχει μια μοναδική ετικέτα, όπως για παράδειγμα η «Xk7EuLhsuHKbnjLWkW4yYGNJJ8OnsgwHBvLQ». Ενώ ένα MediaStream αποτελείται από ένα ή περισσότερα MediaStreamTracks. Κάθε MediaStreamTrack έχει ένα τύπο, video ή audio και μια ετικέτα και αντιπροσωπεύει ένα ή περισσότερα κανάλια video και ήχου αντίστοιχα. Η προσπέλαση των MediaStreamTracks, ανάλογα με τον τύπο τους γίνεται με τις συναρτήσεις getAudioTracks() και getVideoTracks(). Πιο συγκεκριμένα η getAudioTracks() επιστρέφει audio MediaStreamTracks ενώ η getVideoTracks() video MediaStreamTracks.

Ένα παράδειγμα χρήσης video και audio MediaStreamTracks μπορεί να είναι μία εφαρμογή συνομιλίας που παίρνει ροές δεδομένων από την μπροστινή και την πίσω κάμερα, το μικρόφωνο και διαμοιράζει μεταξύ των χρηστών την οθόνη τους.

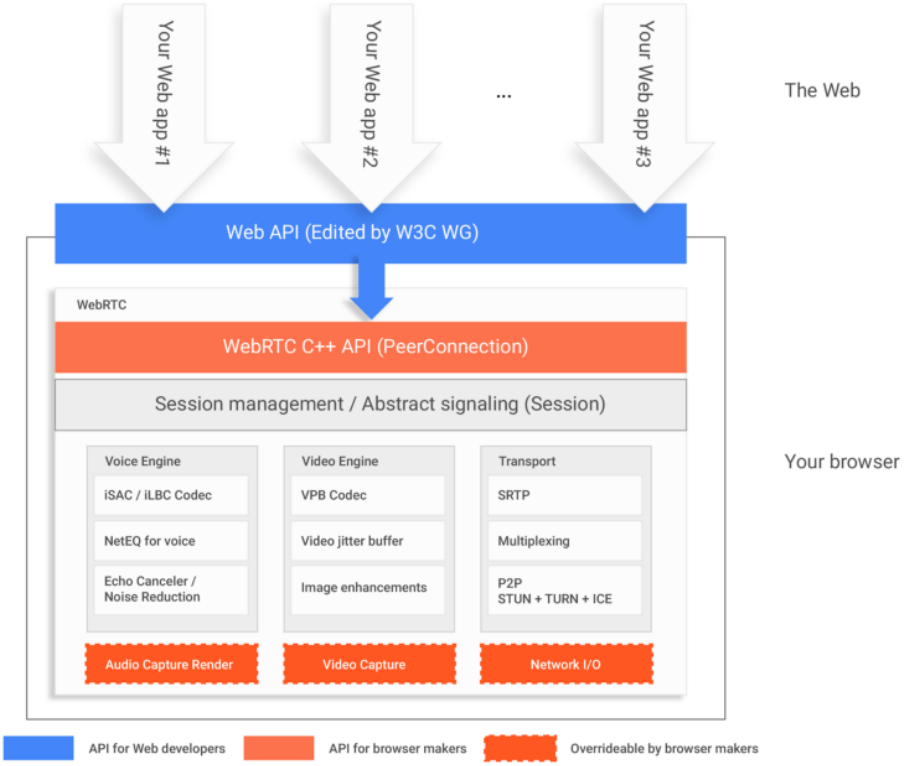
Ένα MediaStreamTrack μπορεί να μοιράζεται μεταξύ πολλών συμμετεχόντων σε μία συνομιλία. Έτσι, όταν ένα ήδη υπάρχων MediaStreamTrack αποστέλλεται σε έναν άλλο συνομιλητή θα εμφανιστεί ως ένα ενιαίο MediaStreamTrack στον παραλήπτη. Για να είναι αυτό δυνατόν θα πρέπει ο παραλήπτης να υποστηρίζει τη προδιαγραφή αυτή. Ο αποστολέας με τη σειρά του, μπορεί να υποδείξει το αντικείμενο MediaStream στο οποίο ανήκει το συγκεκριμένο MediaStreamTrack. Αντίστοιχα, ο παραλήπτης θα πρέπει να δημιουργήσει αντίστοιχα αντικείμενα MediaStream στην πλευρά του, εάν δεν υπάρχουν ήδη, τα οποία θα πρέπει να συμπληρωθούν ανάλογα.

Στους Chrome και Opera, χρησιμοποιείται η μέθοδος URL.createObjectURL() η οποία μετατρέπει ένα MediaStream σε μια URL Blob που μπορεί να οριστεί ως src ενός στοιχείου video. Αυτό γίνεται καθώς και οι δύο browsers επιτρέπουν τη μετάδοση δεδομένων ήχου, μέσω της getUserMedia() σε ένα στοιχείο ήχου ή βίντεο. Επιπλέον αυτοί οι δύο browsers, έχουν την δυνατότητα αποθήκευσης των δικαιωμάτων άδειας ήχου (audioCapture permissions) ή / και video (videoCapture permissions) και χρειάζεται μόνο η αίτηση για χορήγηση άδειας από τον χρήστη μόνο κατά την εγκατάσταση της εφαρμογής. Έτσι, στη συνέχεια, δεν ζητείται άδεια από τον χρήστη για πρόσβαση σε κάμερα ή / και μικρόφωνο.

Σκοπός είναι στο μέλλον να μπορεί να ενεργοποιηθεί ένα MediaStream για κάθε πηγή δεδομένων, όχι μόνο μια camera ή μικρόφωνο. Αυτό θα επιτρέπει τη μετάδοση ροών από το δίσκο ή από αυθαίρετες πηγές δεδομένων όπως αισθητήρες ή και άλλες εισόδους.

RTCPeerConnection

Το RTCPeerConnection API είναι το στοιχείο του WebRTC που παρέχει σταθερή και αποτελεσματική επικοινωνία με χρήση ροών δεδομένων μεταξύ των συνομιλητών. Μπορούμε να δούμε τον ρόλο του RTCPeerConnection, στο ακόλουθο διάγραμμα, όπου φαίνεται η αρχιτεκτονική του WebRTC που φαίνεται και ο αντίστοιχος ρόλος του RTCPeerConnection.



Εικόνα 22: Αρχιτεκτονική WebRTC[10]

Όπως φαίνεται και από το παραπάνω διάγραμμα, το πράσινο κομμάτι, που αποτελεί τον browser είναι αρκετά πολύπλοκο. Ο ρόλος, λοιπόν, του RTCPeerConnection σε συνδυασμό με την χρήση της JavaScript είναι να κρύβουν από τους προγραμματιστές ιστού όλη την πολυπλοκότητα που υπάρχει στο υπόβαθρο. Πιο συγκεκριμένα, οι κωδικοποιητές και τα πρωτόκολλα που χρησιμοποιεί το WebRTC κάνουν μια σημαντική δουλειά, αποκρύπτοντας όλη την περιττή πολυπλοκότητα, για να είναι δυνατή η επικοινωνία σε πραγματικό χρόνο, ακόμη και σε αναξιόπιστα δίκτυα. Με τον όρο αναξιόπιστο δίκτυο, αναφερόμαστε σε ένα δίκτυο όπου είναι πιθανό να συμβεί κάτι από τα παρακάτω:

- απόκρυψη απώλειας πακέτων
- ακύρωση ηχούς
- προσαρμοστικότητα εύρους ζώνης
- δυναμικό jitter buffering

[10] www.webrtc.org

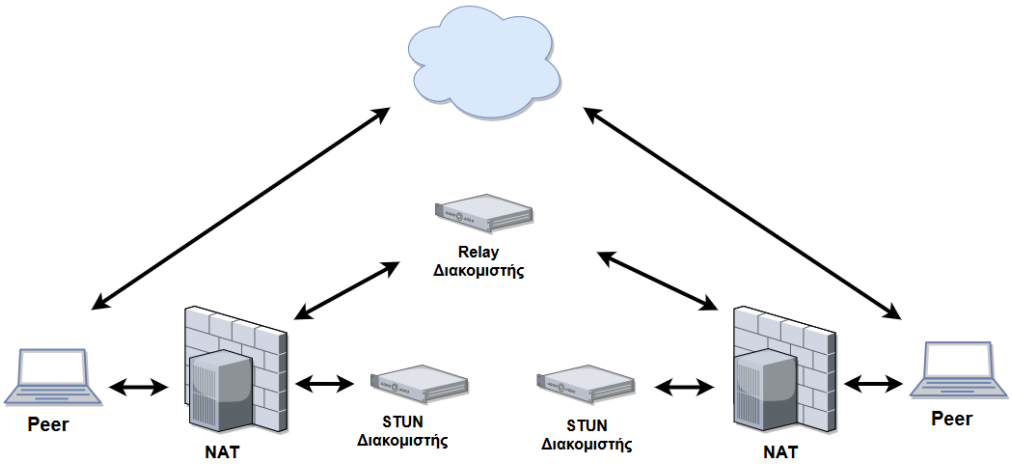
- αυτόματη αναβολή ελέγχου
- μείωση και καταστολή θορύβου
- εικονικός «καθαρισμός»

Όπως έχουμε ήδη αναφέρει το WebRTC χρειάζεται servers για να λειτουργήσει αποτελεσματικά. Πιο συγκεκριμένα, απαιτούνται οι ακόλουθοι τέσσερις servers που προσφέρουν συγκεκριμένες λειτουργίες:

- Ανακάλυψη και επικοινωνία χρηστών. Οι χρήστες ανακαλύπτουν ο ένας τον άλλον και ανταλλάσσουν λεπτομέρειες, όπως τα ονόματά τους.
- Σηματοδότηση. Οι εφαρμογές πελάτη WebRTC (peers) ανταλλάσσουν πληροφορίες δικτύου.
- NAT / firewall. Οι εφαρμογές πελάτη WebRTC διαπερνούν πύλες NAT και τείχη προστασίας.
- Server αναμετάδοσης σε περίπτωση αποτυχίας επικοινωνίας μεταξύ των συνομιλητών. Οι συνεργάτες ανταλλάσσουν πληροφορίες σχετικά με τα δεδομένα που πρόκειται να ανταλλαχθούν, όπως η μορφή και η ανάλυση ενός video.

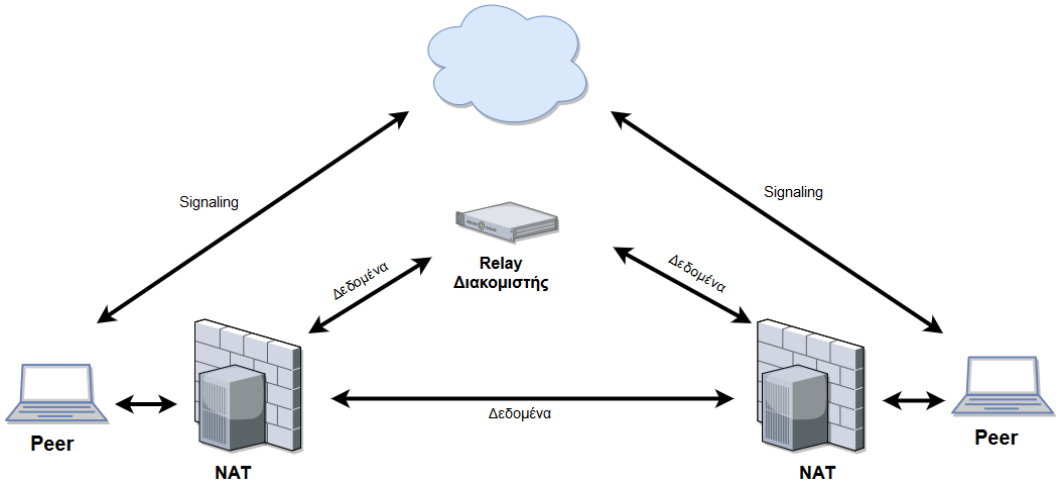
Το WebRTC χρησιμοποιεί το πρωτόκολλο STUN και την επέκτασή αυτού, το TURN στο ICE framework για να επιτρέψουν στο RTCPeerConnection να μπορεί να επικοινωνεί με το NAT traversal και άλλα πρωτόκολλα δικτύου.

Το ICE είναι ένα framework για τη σύνδεση των συνομιλητών, όπως για παράδειγμα δύο χρήστες που συνομιλούν μέσω video. Αρχικά, το ICE προσπαθεί να συνδέσει τους δύο συνομιλητές απευθείας, με τη χαμηλότερη δυνατή καθυστέρηση, χρησιμοποιώντας UDP. Σε αυτή τη διαδικασία, οι STUN server είναι υπεύθυνοι να βοηθήσουν τον κάθε συνομιλητή να βρει την IP διεύθυνση και τη θύρα του, πίσω από ένα NAT.



Εικόνα 23: Απευθείας σύνδεση συνομιλητών με UDP

Εάν το UDP αποτύχει, το ICE δοκιμάζει με TCP: πρώτα με HTTP και στη συνέχεια με HTTPS. Συνήθως η άμεση σύνδεση μπορεί να αποτύχει, κυρίως λόγω των δικτύων NAT και των τειχών προστασίας. Έτσι, το ICE χρησιμοποιεί έναν ενδιάμεσο TURN server αναμετάδοσης. Με άλλα λόγια, το ICE θα χρησιμοποιήσει πρώτα STUN με UDP για να συνδέσει απευθείας τους συνομιλητές και, αν αποτύχει, θα χρησιμοποιήσει τον TURN server.



Εικόνα 24: Σύνδεση συνομιλητών με TCP

RTCDataChannel

Εκτός από ήχο και video, το WebRTC υποστηρίζει την επικοινωνία σε πραγματικό χρόνο και για άλλους τύπους δεδομένων. Το RTCDataChannel API επιτρέπει ανταλλαγή από peer-to-peer χρήστες αυθαίρετων δεδομένων, με χαμηλή λανθάνουσα κατάσταση και υψηλή απόδοση.

Υπάρχουν πολλές περιπτώσεις χρήσης του API αυτού, όπως:

- Παιχνίδια
- Απομακρυσμένες εφαρμογές γραφείου
- Συνομιλία μέσω κειμένου σε πραγματικό χρόνο
- Μεταφορά αρχείων
- Αποκεντρωμένα δίκτυα

Έτσι, το RTCDataChannel API διαθέτει πολλά χαρακτηριστικά για να αξιοποιήσει στο έπακρο το RTCPeerConnection και να επιτρέψει ισχυρή και ταυτόχρονα ευέλικτη peer-to-peer επικοινωνία. Πιο συγκεκριμένα, το RTCDataChannel API μπορεί να:

- Αξιοποιεί την εγκατάσταση της συνεδρίας μέσω του RTCPeerConnection
- Υποστηρίζει πολλαπλά ταυτόχρονα κανάλια επικοινωνίας, με προτεραιότητα
- Υποστηρίζει αξιόπιστη και αναξιόπιστη παράδοση semantics
- Διαθέτει ενσωματωμένη ασφάλεια (DTLS) και έλεγχο συμφόρησης
- Χρησιμοποιηθεί με ή χωρίς ήχο ή βίντεο

Η επικοινωνία ανάμεσα στα προγράμματα περιήγησης μπορεί να γίνει απευθείας, οπότε το RTCDataChannel μπορεί να είναι πολύ ταχύτερο από το WebSocket, ακόμη και αν απαιτείται ένας διακομιστής αναμετάδοσης (TURN), όταν η επικοινωνία με χρήση τειχών προστασίας και NAT αποτύχει.

5. Το σύστημα στην πράξη

5.1. Είσοδος χρήστη

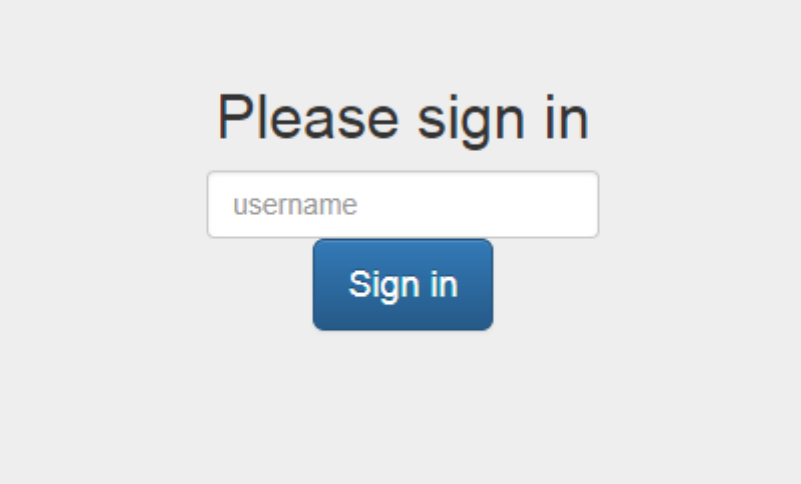
Για να μπορέσει κάποιος χρήστης να λάβει μέρος στην επικοινωνία και να πραγματοποιήσει μία κλήση, θα πρέπει αρχικά να συνδεθεί στο σύστημα χρησιμοποιώντας το όνομά του. Εναλλακτικά, ορισμένα συστήματα WebRTC, δίνουν την δυνατότητα στους χρήστες να συνδέονται σε αυτό χρησιμοποιώντας ένα μοναδικό id. Το στοιχείο που θα χρησιμοποιήσει τελικά ο χρήστης για συνδεθεί στο σύστημα αποτελεί την ταυτότητά του σε αυτό. Πιο συγκεκριμένα, με το στοιχείο αυτό ο χρήστης αποθηκεύεται, καλείται και τελικά αποσυνδέεται από το σύστημα όταν αυτό είναι απαραίτητο.

Κάθε φορά που ένας χρήστης συνδέεται στο σύστημα, ένα μήνυμα register φεύγει από τον client προς τον sever με το επιλεγμένο όνομα από τον χρήστη. Για να μπορέσει ο server να διαχειριστεί αυτά τα μηνύματα, υπάρχει η κλάση UserRegistry, η οποία σχετίζεται με την αποθήκευση και τον εντοπισμό των χρηστών. Η κλάση UserRegistry καλείται κάθε φορά που ένα μήνυμα register φτάνει στον server ώστε να διαπιστωθεί ότι το όνομα του χρήστη που μόλις προσπάθησε να συνδεθεί είναι μοναδικό και δεν χρησιμοποιείται από κάποιον άλλον χρήστη. Στην περίπτωση που το όνομα που έχει επιλέξει ο χρήστης, χρησιμοποιείται ήδη, δεν επιτρέπεται στον χρήστη η είσοδός του στο σύστημα.

Εφόσον ο χρήστης συνδεθεί στο σύστημα, προστίθεται στην λίστα των συνδεδεμένων χρηστών. Κατ' επέκταση, εμφανίζεται στη λίστα των συνδεδεμένων χρηστών σε όλους τους χρήστες που είναι ήδη συνδεδεμένοι στο σύστημα. Κατ' αυτό τον τρόπο, οι υπόλοιποι χρήστες μπορούν, στην συνέχεια, να επικοινωνήσουν μαζί του.

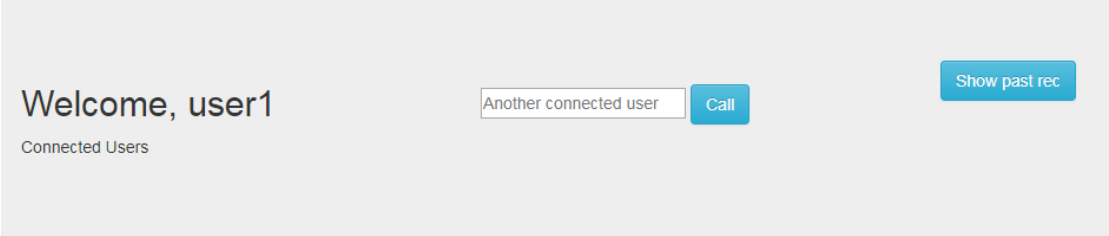
Για κάθε χρήστη που συνδέεται στο σύστημα καλείται η συνάρτηση register(), από την πλευρά του χρήστη, όταν ληφθεί το μήνυμα register. Η συνάρτηση αυτή δημιουργείται ένα UserSession, καλώντας την κλάση UserSession. Το UserSession χρησιμοποιείται για την συνεχή επικοινωνία μεταξύ client και server με τη χρήση αιτήσεων (request) και απαντήσεων (response). Τέλος, σκοπός της κλάσης UserSession είναι η δέσμευση και απελευθέρωση των πόρων στον Kurento Media Server κατά την αποθήκευση μίας συνεδρίας του χρήστη.

Στην εικόνα 25 φαίνεται η φόρμα σύνδεσης του χρήστη. Όπως έχουμε ήδη αναφέρει ένας χρήστης αρκεί να συμπληρώσει το όνομά του στο κενό πεδίο username και στη συνέχεια, να πατήσει στο πλήκτρο «Sign in» για να μπορέσει να συνδεθεί στο σύστημα.



Εικόνα 25: Φόρμα σύνδεσης χρήστη

Για να είναι πετυχημένη η σύνδεση ενός χρήστη στο σύστημα, ο server στέλνει ένα μήνυμα registered προς τον client. Έτσι, ο χρήστης συνδέεται στο σύστημα και εμφανίζεται σε αυτόν ένα μήνυμα που τον καλωσορίζει. Το μήνυμα αυτό έχει την μορφή «Welcome,» ακολουθούμενο από το όνομα που έχει επιλέξει ο χρήστης για τη σύνδεσή του στο σύστημα, όπως φαίνεται και στην ακόλουθη εικόνα.



Εικόνα 26: Με την είσοδο του χρήστη στο σύστημα εμφανίζεται ένα μήνυμα καλωσορίσματος

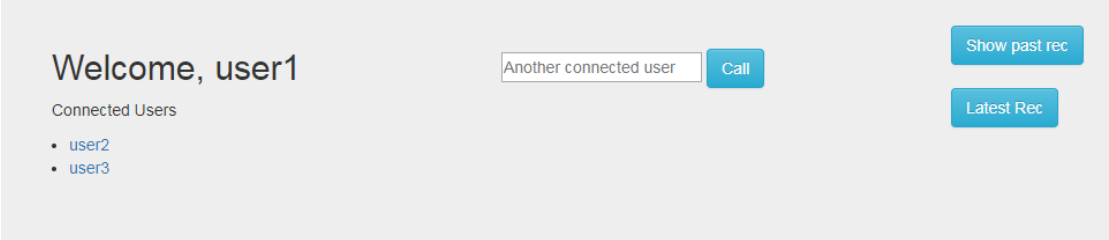
5.2. Λίστα συνδεδεμένων χρηστών

Για να είναι δυνατή η επικοινωνία των χρηστών του συστήματος, το WebRTC εμφανίζει σε κάθε χρήστη μία λίστα με όλους τους υπόλοιπους συνδεδεμένους χρήστες στο σύστημα. Η λίστα αυτή προκύπτει από τη μεταβλητή registeredUsers, η οποία βρίσκεται στην πλευρά του server. Η λίστα των συνδεδεμένων χρηστών ενημερώνεται, προσθέτοντας ένας νέο χρήστη σε αυτή, όταν ένας χρήστης συνδεθεί επιτυχώς στο σύστημα ή αφαιρώντας ένας χρήστη από αυτή, όταν ο αντίστοιχος χρήστης αποσυνδεθεί από στο σύστημα.

Η λίστα αυτή ενημερώνεται καλώντας τη κλάση UserRegistry που αναφέραμε και προηγουμένως. Ενώ, κάθε φορά που ενημερώνεται φεύγει ένα μήνυμα από τον server προς τον client με σκοπό την ανανέωση της λίστας. Πιο συγκεκριμένα, φεύγουν τα μηνύματα addUsers και removeUsers για την προσθήκη και τη διαγραφή χρηστών από τη λίστα αντίστοιχα. Ο client με τη σειρά του, ενημερώνει όλους τους συνδεδεμένους χρήστες για τους υπόλοιπους συνδεδεμένους χρήστες στο σύστημα. Ανάλογα με την ενέργεια που έχει προηγηθεί, η λίστα που βλέπει ο χρήστης είτε ενημερώνεται προσθέτοντας είτε αφαιρώντας άτομα από αυτή.

Έτσι, μόλις ένα χρήστης συνδεθεί στο σύστημα, βλέπει όλους τους υπόλοιπους συνδεδεμένους χρήστες σε αυτό σε μία λίστα κάτω από το Connected Users. Επιλέγοντας κάποιον από αυτούς μπορεί να τους καλέσει και να ξεκινήσει μία συνδιάσκεψη μεταξύ τους.

Στη συνέχεια, φαίνεται η αντίστοιχη εικόνα, όπου ο χρήστης user1 βλέπει δύο συνδεδεμένους χρήστες, τους χρήστες user2 και user3. Για να μπορέσει να επικοινωνήσει με κάποιον από τους δύο χρήστες αρκεί να επιλέξει το όνομά του και στη συνέχεια να πατήσει το κουμπί «Call».



Εικόνα 27: Λίστα συνδεδεμένων χρηστών

5.3 Εκκίνηση επικοινωνίας

Στο σύστημα, για να είναι δυνατή η επικοινωνία θα πρέπει να έχουν συνδεθεί τουλάχιστον δύο χρήστες, ο καλών (caller) όσο και ο καλούμενος (callee). Η επικοινωνία μεταξύ των διάφορων χρηστών, θα ξεκινήσει μόνο όταν αυτοί συνδεθούν σε ένα κοινό δωμάτιο.

Κάθε δωμάτιο που δημιουργείται, είναι απομονωμένο και ανεξάρτητο από όλα τα άλλα δωμάτια. Έτσι, οι χρήστες που συνδέονται σε ένα συγκεκριμένο δωμάτιο, μπορούν να ανταλλάσσουν δεδομένα μόνο μεταξύ τους και όχι με χρήστες άλλων δωματίων. Κάθε χρήστης θα στείλει τα δικά του μηνύματα και με τη σειρά του θα λάβει μηνύματα από όλους τους άλλους συμμετέχοντες στο ίδιο δωμάτιο. Αυτό σημαίνει ότι θα υπάρξουν συνολικά n*n webrtc τελικά σημεία επικοινωνίας σε κάθε δωμάτιο, όπου n είναι ο αριθμός των συμμετεχόντων σε αυτό.

Η εκκίνηση της επικοινωνίας μεταξύ δύο χρηστών ξεκινάει όταν ο ένας από τους δύο, επιλέξει τον άλλον από τη λίστα των συνδεδεμένων χρηστών και στη συνέχεια πατήσει το κουμπί «Call». Στην ακόλουθη εικόνα, ο χρήστης user1 έχει επιλέξει το χρήστη user2 με σκοπό την εκκίνηση της επικοινωνίας.

Εικόνα 28: Εκκίνηση επικοινωνίας μεταξύ των χρηστών user1 και user2.

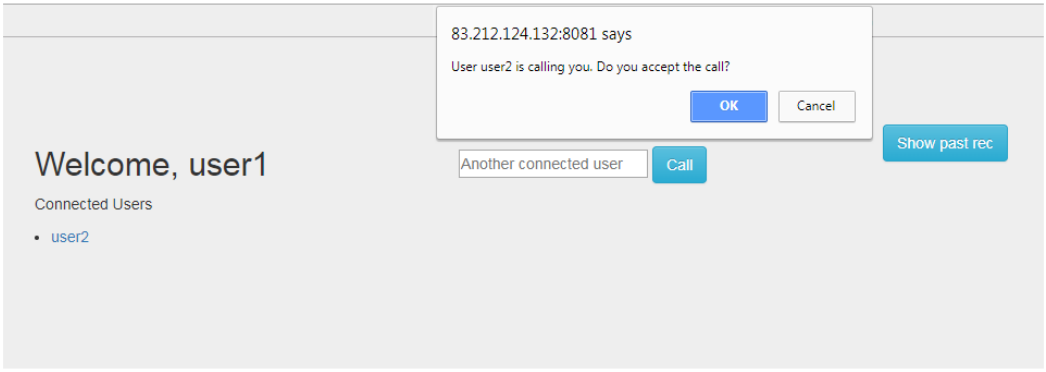
Σε αυτή την περίπτωση, θα σταλθεί ένα μήνυμα call, για το αίτημα της κλήση προς το χρήστη αυτόν, από τον client προς τον server και θα κληθεί η συνάρτηση call().

Η συνάρτηση call() αρχικά ελέγχει ότι ο χρήστης που πάτησε το κουμπί «Call», δεν έχει καλέσει τον εαυτό του. Στην περίπτωση που ένας χρήστης καλεί τον εαυτό του, η κλήση απορρίπτεται και ο χρήστης ενημερώνεται με αντίστοιχο μήνυμα για αυτό. Στη συνέχεια, η συνάρτηση ελέγχει ότι ο χρήστης που καλείται δεν είναι κάποιος με τον οποίο ο καλών βρίσκεται ήδη σε επικοινωνία. Αυτό το ενδεχόμενο είναι πιθανό, μόνο όταν ο καλών και ο καλούμενος βρίσκονται ήδη σε ένα κοινό δωμάτιο επικοινωνίας. Και σε αυτή την περίπτωση, ο καλών ενημερώνεται με αντίστοιχο μήνυμα. Πριν ξεκινήσει η επικοινωνία, γίνονται κάποιοι επιπλέον έλεγχοι, όπως για παράδειγμα αν ο καλούμενος είναι κάποιος από τους εγγεγραμμένους χρήστες.

Εφόσον, πετύχουν όλοι οι προηγούμενοι έλεγχοι, δημιουργείται το όνομα του δωματίου στο οποίο θα συμμετάσχουν οι δύο χρήστες (καλών και καλούμενος). Το όνομα του δωματίου που δημιουργείται είναι μοναδικό και αποτελεί ένα συνδυασμό ενός τυχαίου αριθμού και της τρέχουσας ώρας. Αν ένας από τους καλών ή καλούμενο, βρίσκεται ήδη σε κάποιο δωμάτιο επικοινωνίας με κάποιον τρίτον χρήστη, τότε θα χρησιμοποιηθεί το όνομα του ήδη υπάρχοντος δωματίου.

Στη συνέχεια, φεύγει μήνυμα incomingCall από τον server προς τον client. Ενώ, ο client με τη σειρά του, θα καλέσει τη συνάρτηση incomingCall(). Μέσω της συνάρτησης αυτής ενημερώνεται ο καλούμενος ότι ένας άλλος χρήστης τον καλεί. Σε αυτή την περίπτωση, ο καλούμενος έχει τη δυνατότητα να δει το όνομα του χρήστη που τον καλεί. Ο καλούμενος έχει τη δυνατότητα είτε αποδοχής είτε απόρριψης της εισερχόμενης κλήσης.

Η ακόλουθη εικόνα αποτελεί ένα παράδειγμα, όπου ο χρήστης user1 καλείται από τον χρήστη user2. Ο χρήστης user1, έχει τη δυνατότητα είτε αποδοχής είτε απόρριψης της κλήσης επιλέγοντας τα κουμπιά «Ok» και «Cancel» αντίστοιχα.



Εικόνα 29: Ο χρήστης user1 δέχεται κλήση από τον χρήστη user2

Στην περίπτωση που ο καλούμενος δεχτεί την κλήση (κουμπί «Ok»), για να είναι εφικτή η εκκίνηση της επικοινωνίας, φεύγει ένα μήνυμα joinRoom από τον client προς τον server. Ο server, με τη σειρά του καλεί τη συνάρτηση joinRoom(), η οποία δέχεται σαν όρισμα το όνομα του δωματίου στο οποίο πρόκειται να συνδεθούν οι δύο χρήστες. Στη συνέχεια, η συνάρτηση joinRoom() καλεί την συνάρτηση getroom() με όρισμα το δοσμένο όνομα δωματίου. Με βάση αυτό το όνομα, δημιουργείται ένα Media Pipeline, το οποίο αποτελείται από ένα στοιχείο, το composite, για τον kurento server.

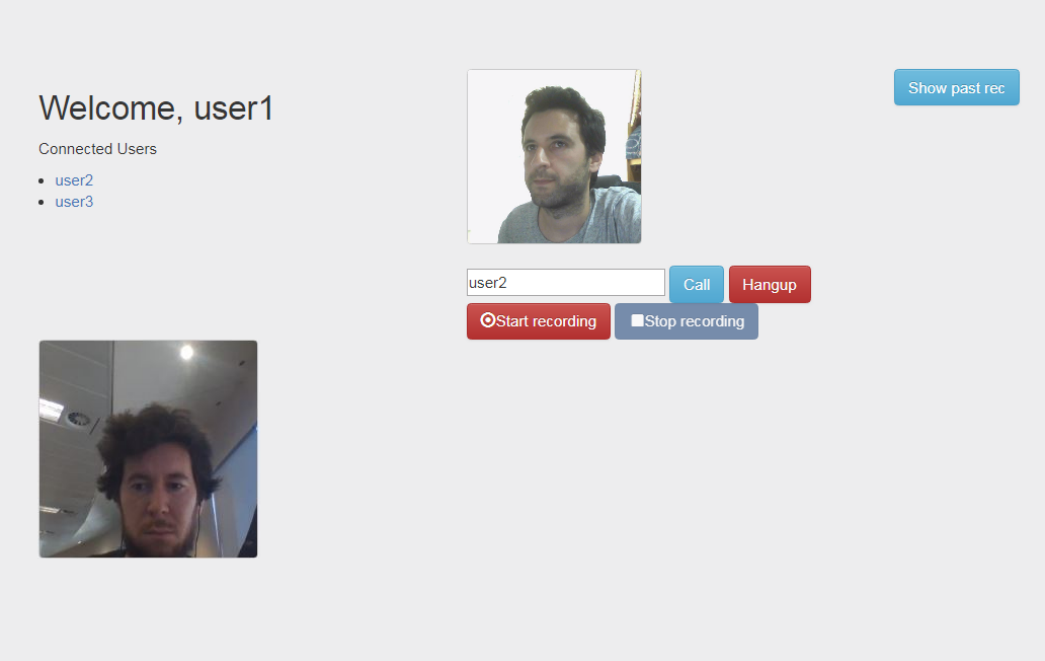
Με τον όρο «Media Pipeline» αναφερόμαστε σε αλυσίδα στοιχείων, όπου το αποτέλεσμα της επεξεργασίας της εισόδου ενός στοιχείου (πηγή), δηλαδή η έξοδός του, τροφοδοτεί ένα ή περισσότερα επόμενα στοιχεία. Με άλλα λόγια αποτελεί την είσοδό τους. Ως εκ τούτου, το «Media Pipeline» αποτελεί έναν αγωγό και αντιπροσωπεύει ένα "μηχάνημα" που είναι ικανό να εκτελέσει μια ακολουθία λειτουργιών σειριακά, τη μία μετά την άλλη.

Τέλος, η συνάρτηση joinRoom() καλεί τη συνάρτηση join(), όπου δημιουργείται το Webrtcendpoint των χρηστών που πρόκειται να συνδεθούν στο δωμάτιο. Το Webrtcendpoint είναι το τελικό σημείο επικοινωνίας σε μία σύνδεση peer-to-peer

WebRTC. Πλέον οι χρήστες μπορούν να συνδεθούν στο δωμάτιο, ενώ οι υπόλοιποι συμμετέχοντες σε αυτό, αν υπάρχουν, ενημερώνονται για την συμμετοχή των νέων χρηστών. Τέλος, οι νέοι συμμετέχοντες στο δωμάτιο ενημερώνονται για όλους τους χρήστες που συμμετέχουν ήδη σε αυτό.

Στη συνέχεια και για να είναι δυνατή η λήψη βίντεο από όλους τους συμμετέχοντες στην συνδιάσκεψη, στέλνεται ένα μήνυμα receiveVideoFrom από τον client προς τον server. Ο server με τη σειρά του καλεί την συνάρτηση receiveVideoFrom(). Μέσω αυτής της συνάρτησης καλείται η συνάρτηση getEndpointForUser() όπου δημιουργούνται τα Webrtcendpoint για την λήψη media μεταξύ των χρηστών. Όπως έχουμε ήδη αναφέρει εκεί συνδέονται οι ροές στο Kurento Media Server και είναι δυνατή η λήψη video μεταξύ των συμμετεχόντων στην συνδιάσκεψη. Τέλος, ο server στέλνει ένα μήνυμα προς τον client receiveVideoAnswer για την επιβεβαίωση της λήψης video μεταξύ των συμμετεχόντων.

Στην περίπτωση που ο καλούμενος απορρίψει την κλήση, φεύγει ένα μήνυμα callResponse από τον client προς τον server. Σε αυτή την περίπτωση, διαγράφεται το μήνυμα που έφυγε από τον καλών προς των καλούμενο σχετικά με την κλήση.



Εικόνα 30: Εκκίνηση επικοινωνίας μεταξύ των χρηστών user1 και user2.

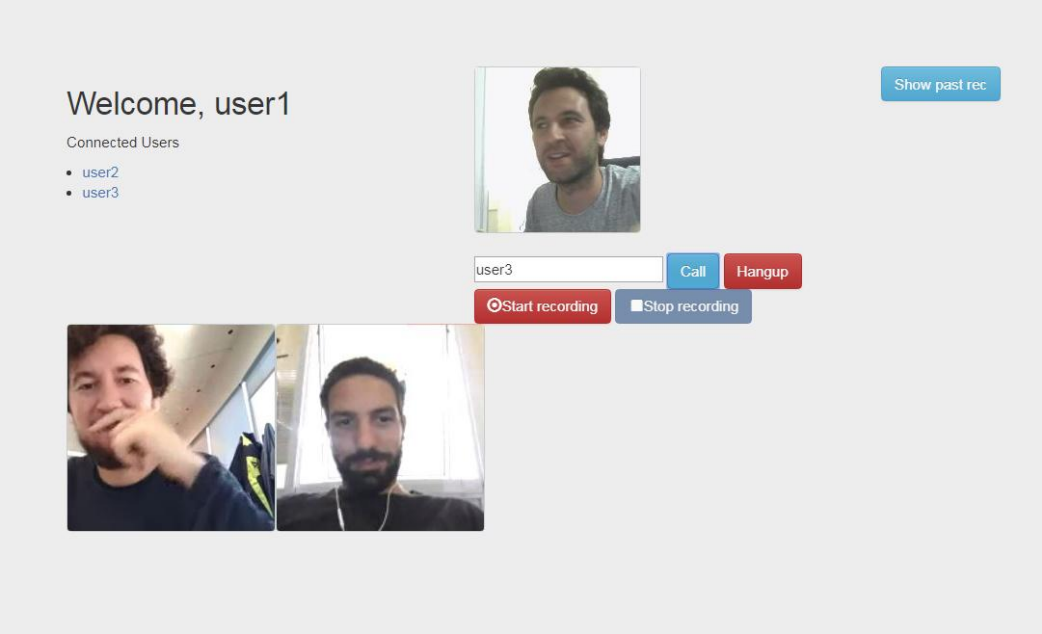
5.4 Προσθήκη συμμετεχόντων στη συνδιάσκεψη

Σε ένα ήδη δημιουργημένο δωμάτιο μπορούν να προστεθούν περισσότεροι χρήστες από τους ήδη υπάρχοντες, κατά τη διάρκεια της συνδιάσκεψης. Για να είναι αυτό εφικτό, κάποιος από τους συμμετέχοντες στο δωμάτιο επικοινωνίας θα πρέπει να καλέσει το νέο αυτό χρήστη.

Πιο συγκεκριμένα, αποτελεί μία υποπερίπτωση της εκκίνησης επικοινωνίας, αφού το δωμάτιο επικοινωνίας υπάρχει ήδη και μία συνδιάσκεψη βρίσκεται σε εξέλιξη. Και σε αυτή την περίπτωση στέλνεται ένα μήνυμα call από τον client προς τον server και ακολουθείται η ίδια ροή μηνυμάτων όπως την εκκίνηση επικοινωνίας.

Σε αυτή την περίπτωση, δημιουργείται ένα νέο Webrtcendpoint για το νέο χρήστη που πρόκειται να εισέλθει στο δωμάτιο επικοινωνίας. Σκοπός είναι, ο νέος χρήστης να λαμβάνει δεδομένα τόσο από τον server όσο και από τους υπόλοιπους συμμετέχοντες στο σωμάτιο. Από την άλλη, όλοι οι ήδη συμμετέχοντες στο δωμάτιο, ενημερώνονται ότι ένας νέος χρήστης έχει συνδεθεί και θα ζητήσουν από τον server να λαμβάνουν δεδομένα και από αυτό τον νέο συμμετέχοντα. Τέλος, ο νέος χρήστης, με τη σειρά του, παίρνει μια λίστα με όλους τους συνδεδεμένους χρήστες στο δωμάτιο και ζητά από τον server να λαμβάνει όλα τα δεδομένα από όλους τους συμμετέχοντες στο δωμάτιο.

Στην ακόλουθη εικόνα φαίνεται ένα δωμάτιο με τρεις συμμετέχοντες. Για να συνδεθεί ένας τρίτος συμμετέχοντας, όπως για παράδειγμα ο χρήστης user3 αρκεί να επιλεγεί το όνομά του και στη συνέχεια να πατηθεί το κουμπί «Call».



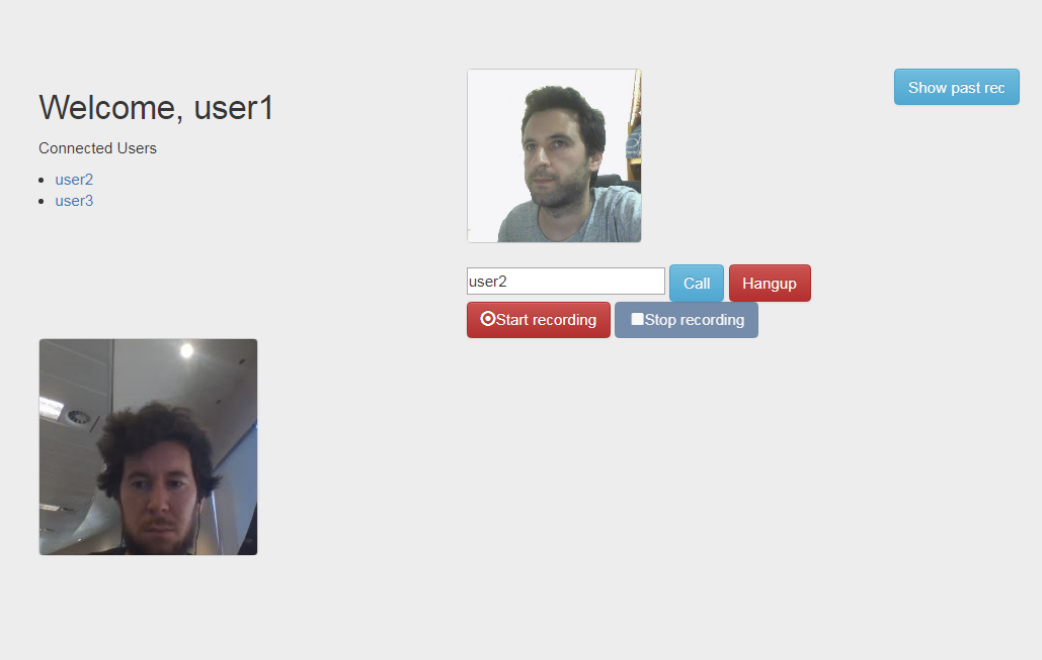
Εικόνα 31: Προσθήκη του χρήστη user3 στο δωμάτιο συνδιάσκεψης των χρηστών user1 και user2

5.5. Καταγραφή συνδιάσκεψης

Για να είναι εφικτή η καταγραφή μίας συνδιάσκεψης, η πληροφορία στέλνεται από το local stream στο Kurento Media Server από όπου επιστρέφει πίσω στον πελάτη ενώ ταυτόχρονα την ίδια στιγμή γίνεται και καταγραφή της πληροφορίας αυτής. Προκειμένου να δημιουργηθεί μια εφαρμογή που καταγράφει ροές WebRTC στο σύστημα αρχείων, πρέπει να δημιουργηθεί ένα «Media Pipeline», το οποίο αποτελείται από δύο στοιχεία, ένα WebRtcEndpoint και ένα RecorderEnpoint.

Έτσι, όταν ένας χρήστης συνδέεται με την εφαρμογή, θα πρέπει να αρχικοποιούνται αυτά τα στοιχεία για να είναι δυνατή η λήψη δεδομένων από το WebRtcEndpoint, που είναι ικανό να λαμβάνει ροές WebRTC προς το RecorderEndpoint, το οποίο είναι ικανό να καταγράφει ροές δεδομένων στο σύστημα αρχείων.

Για να ξεκινήσει η καταγραφή μίας συνδιάσκεψης, θα πρέπει κάποιος από τους συνδεδεμένους χρήστες σε ένα δωμάτιο να πατήσει το κουμπί «Start recording», όπως φαίνεται στην ακόλουθη εικόνα.



Εικόνα 32: Καταγραφή συνδιάσκεψης με το κουμπί Start recording

Όταν το κουμπί έναρξης καταγραφής πατηθεί, ένα μήνυμα startRecording στέλνεται από τον client προς τον server. Ο server, με τη σειρά του καλεί την συνάρτηση startRecord(), η οποία υλοποιεί όλη την απαιτούμενη λογική της καταγραφής μίας συνδιάσκεψης.

Πιο συγκεκριμένα, πριν ξεκινήσει η καταγραφή μίας συνδιάσκεψης, η συνάρτηση προσπελαύνει το όνομα του δωματίου με βάση το userSession. Στη συνέχεια, ορίζει το όνομα του αρχείου καταγραφής. Τα αρχεία καταγραφής έχουν κατάληξη .webm, ενώ το όνομά τους πρέπει να είναι μοναδικό. Για να είναι το όνομα του αρχείου καταγραφής μοναδικό, ορίζεται ένας μοναδικός συνδυασμός ενός τυχαίου αριθμού και της τρέχουσας ώρας, όπως και στην περίπτωση της ονομασίας του δωματίου επικοινωνίας. Τέλος, όλα τα αρχεία καταγραφής ξεκινούν με τη λέξη record. Συνολικά, ένα αρχείο καταγραφής ονομάζεται με τη λέξη record ακολουθούμενο από ένα μοναδικό κωδικό και στο τέλος έχει την κατάληξη .webm. Τα αρχεία που δημιουργούνται έχουν κατάληξη .webm, ενώ αποθηκεύονται στο URI /home/user/webapp/static/records/.

Τα αρχεία webm είναι αρχεία που μπορούν να περιέχουν συνδυασμό οπτικοακουστικών δεδομένων. Αυτό σημαίνει πως ένα webm αρχείο μπορεί να

περιέχει βίντεο (εικόνα) και ήχο, μόνο βίντεο (εικόνα) ή μόνο ήχο. Τα αρχεία webm αποθηκεύουν βίντεο συμπιεσμένο με τη χρήση της τεχνολογίας VP8 και ήχο συμπιεσμένο με συμπίεση Ogg Vorbis. Χρησιμοποιούνται συνήθως για την παροχή διαδικτυακών βίντεο χρησιμοποιώντας την ετικέτα HTML5. Στην περίπτωση της υλοποίησής μας επιλέξαμε να χρησιμοποιήσουμε τον τύπο αρχείων webm λόγω του ότι:

- Είναι λογισμικό ανοιχτού κώδικα
- Υποστηρίζεται από τους πιο ευρέως διαδεδομένους browser χωρίς την ανάγκη χρήσης επιπροσθέτων
- Είναι πιο «φιλικός» στη χρήση σε εφαρμογές ιντερνέτ

Στη συνέχεια και αφού οριστούν όλα τα παραπάνω στοιχεία, ορίζεται το RecorderEndpoint. Το RecorderEndpoint παρέχει όλη τη λειτουργικότητα για την αποθήκευση περιεχομένου είτε σε τοπικά αρχεία είτε σε πόρους δικτύου. Για να το πετύχει αυτό λαμβάνει σαν είσοδο μια ροή πολυμέσων από ένα διαφορετικό MediaElement, που στην συγκεκριμένη περίπτωση είναι το WebRtcEndpoint που λειτουργεί σαν πηγή και τέλος τα αποθηκεύει σε μία προκαθορισμένη τοποθεσία.

Για να ξεκινήσει η καταγραφή ορίζονται κάποιες επιπλέον παράμετροι, όπως το recordUri που αποτελείται από την διεύθυνση του συστήματος ακολουθούμενο από το όνομα του αρχείου καταγραφής. Επιπλέον κρατείται η ώρα έναρξης της καταγραφής και τέλος οι συμμετέχοντες σε αυτή. Στη συνέχεια, σε κάθε έναν από τους συμμετέχοντες στέλνεται ένα μήνυμα startedRecording με το οποίο ενημερώνονται ότι η καταγραφή της συνδιάσκεψης έχει ξεκινήσει.

Η δομή της βάσης δεδομένων στην οποία αποθηκεύονται τα μεταδεδομένα της καταγραφόμενης συνδιάσκεψης είναι διαθέσιμη στο κεφάλαιο 4.1.1.3

Τα αρχεία ήχου της συνδιάσκεψης δεν αποθηκεύονται στη βάση δεδομένων της καταγραφής (όπως τα μεταδεδομένα που περιγράφουν τη συνδιάσκεψη), αλλά ο Kurento media server τα αποθηκεύει στο filesystem πάνω στο οποίο είναι εγκατεστημένος. Δυνητικά ο Kurento media server μπορεί να ρυθμιστεί ώστε να αποθηκεύει τα εν λόγω αρχεία σε οποιοδήποτε άλλο διαθέσιμο filesystem. Στη βάση δεδομένων αποθηκεύεται το URI με το οποίο ένας χρήστης μπορεί να προσπελαύνει το αρχείο στο μέλλον.

5.6. Τερματισμός καταγραφής

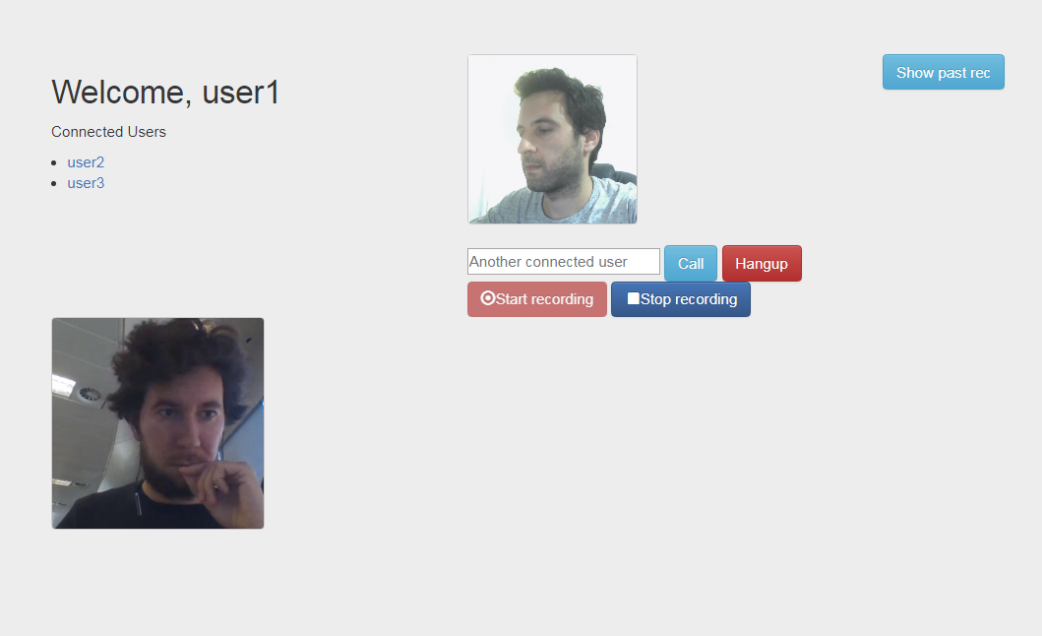
Για να είναι εφικτός, ο τερματισμός της καταγραφής μίας συνδιάσκεψης, θα πρέπει κάποιος από τους συμμετέχοντες να πατήσει το κουμπί «Stop recording», εφόσον προηγουμένως έχει ήδη πατηθεί το πλήκτρο «Start recording». Όταν πατηθεί το κουμπί διακοπής καταγραφής, Stop recording, το στοιχείο εγγραφής, RecorderEndpoint σταματά να καταγράφει και όλα τα στοιχεία απελευθερώνονται. Αυτή είναι μία εργασία για την οποία είναι υπεύθυνος ο server.

Με το κουμπί διακοπής καταγραφής, «Stop recording», ένα μήνυμα stopRecording στέλνεται από τον client προς τον server. Ο server με τη σειρά του, καλεί την συνάρτηση stopRecord(). Όπως ακριβώς και η συνάρτηση startRecord(), πριν τον τερματισμό της καταγραφής μίας συνδιάσκεψης, προσπελαύνει το όνομα του δωματίου με βάση το userSession. Στη συνέχεια, για τον τερματισμό της καταγραφής καλείται η συνάρτηση stop() για το RecorderEndpoint. Επιπλέον, αποθηκεύεται η ώρα τερματισμού της καταγραφής και το αρχείο καταγραφής αποθηκεύεται.

Όλοι οι χρήστες που συμμετείχαν στο συγκεκριμένο δωμάτιο ενημερώνονται ότι η καταγραφή τερματίστηκε, μέσω ενός μηνύματος recordUrl. Στην πραγματικότητα, μέσω του μηνύματος αυτού είναι δυνατή η δημιουργία του κουμπιού της πιο πρόσφατης καταγραφής «Latest rec», όπως θα δούμε στη συνέχεια. Μέσω του κουμπιού «Latest rec», κάθε χρήστης έχει τη δυνατότητα αναπαραγωγής της πιο πρόσφατης καταγραφής.

Όπως έχουμε ήδη αναφέρει κάποιες επιπλέον πληροφορίες αποθηκεύονται μαζί με το αρχείο καταγραφής. Οι πληροφορίες αυτές σχετίζονται με την ώρα έναρξης και λήξης της καταγραφής καθώς και τους συμμετέχοντες στο δωμάτιο επικοινωνίας. Όλες αυτές οι πληροφορίες χρησιμοποιούνται τόσο για την εύρεση του αρχείου καταγραφής και την εμφάνιση των αντίστοιχων κουμπιών στους αντίστοιχους συμμετέχοντες όσο και για να είναι δυνατή η αναπαραγωγή του αρχείου καταγραφής αργότερα.

Στην ακόλουθη εικόνα, ο τερματισμός της καταγραφής της συνδιάσκεψης μπορεί να πραγματοποιηθεί πατώντας το κουμπί «Stop recording»



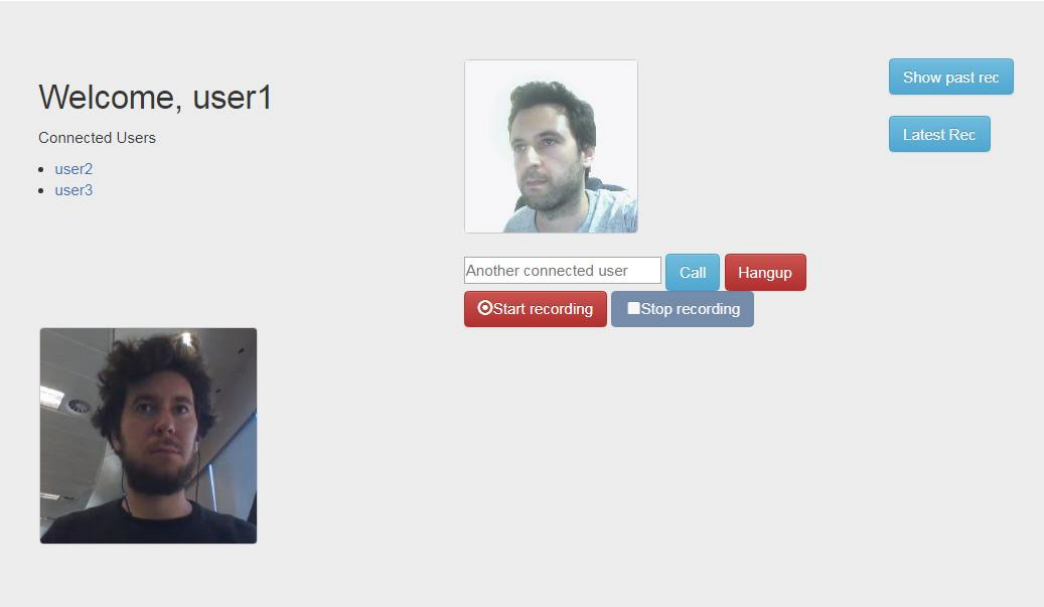
Εικόνα 33: Τερματισμός καταγραφής συνδιάσκεψης με το κουμπί Stop recording

5.7. Τερματισμός συνδιάσκεψης

Ο τερματισμός μιας συνδιάσκεψης γίνεται όταν όλοι οι συμμετέχοντες σε ένα δωμάτιο απομακρυνθούν από αυτό. Ωστόσο η απελευθέρωση όλων των πόρων της επικοινωνίας δεν γίνεται από έναν οποιοδήποτε από τους συμμετέχοντες αλλά από τον server. Για να απομακρυνθεί ένας χρήστης από τη συνδιάσκεψη, θα πρέπει να πατήσει το κουμπί «Hangup».

Όταν το κουμπί «Hangup» πατηθεί ένα μήνυμα leaveRoom φεύγει από τον client προς τον server. Ο server με τη σειρά του καλεί τη συνάρτησης leaveRoom(), μέσω της οποίας ενημερώνει όλους τους συμμετέχοντες στο συγκεκριμένο δωμάτιο ότι ένας χρήστης αποχωρεί από αυτό. Πιο συγκεκριμένα, οι χρήστες ενημερώνονται και για το ποιος χρήστης αποχωρεί. Στη συνέχεια, ενημερώνεται η λίστα με τους συμμετέχοντες στο συγκεκριμένο δωμάτιο ενώ παράλληλα γίνονται release όλα τα resources του χρήστη αυτού. Με αυτό τον τρόπο, οι υπόλοιποι εναπομείναντες συμμετέχοντες δεν λαμβάνουν κάποια πληροφορία από τον χρήστη που έφυγε. Οι υπόλοιποι συμμετέχοντες στο δωμάτιο ενημερώνονται από το μήνυμα participantLeft, το οποίο στέλνεται από τον server προς τον client.

Με την απομάκρυνση κάθε χρήστη, η συνάρτηση leaveRoom() ελέγχει κατά πόσο στο δωμάτιο έχουν μείνει εναπομείναντες χρήστες. Εφόσον στον δωμάτιο δεν υπάρχει πλέον κανένας συμμετέχοντας, καταργείται το αντίστοιχο Media Pipeline γίνεται release και το δωμάτιο διαγράφεται. Για να συμβεί αυτό θα πρέπει στο δωμάτιο να έχουν απομείνει δύο συμμετέχοντες και ένας από αυτούς να πατήσει το κουμπί «Hangup».



Εικόνα 34: Τερματισμός συνδιάσκεψης με το κουμπί Hangup.

Εξετάζοντας τα στατιστικά του όγκου δεδομένων κατά τη διάρκεια πραγματοποίησης μιας συνδιάσκεψης καταγράφουμε τα αποτελέσματα που παρουσιάζονται στον πίνακα 1. Ο πίνακας παρουσιάζει το σύνολο των δεδομένων που μεταφέρονται (download/upload) απο έναν χρηστη στις περιπτώσεις συνδιάσκεψης με 2 ή 3 συμμετέχοντες και στις περιπτώσεις διάρκειας 2, 3 και 5 λεπτών.

Χρηστες	Διάρκεια (λεπτά)		
	1	3	5
2	4.35 Mb	11.2 Mb	19.4 Mb
3	5.62 Mb	14.53 Mb	28.01 Mb

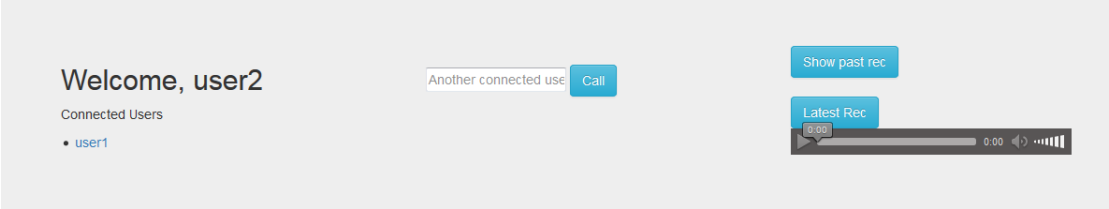
Πίνακας 1: Όγκος δεδομένων συνδιάλεξης

Παρατηρώντας τις τιμές του παραπάνω πίνακα βλέπουμε ότι ανάλογα με τον αριθμό των χρηστών ο όγκος δεδομένων αυξάνεται αναλογικά. Το ίδιο συμβαίνει και αντίστοιχα για την περίπτωση της διάρκειας της συνδιάσκεψης.

5.8. Αναπαραγωγή τελευταίας καταγραφής

Με τον τερματισμό μίας καταγραφής, στην οθόνη του χρήστη εμφανίζεται ένα κουμπί, με την ονομασία «Latest Rec», με το οποίο πατώντας το ο χρήστης δημιουργείται ένα αντικείμενο audio για την αναπαραγωγή του πιο πρόσφατου αρχείου καταγραφής. Για να είναι δυνατή η αναπαραγωγή του αρχείου αυτού, όπως θα δούμε και παρακάτω απαιτείται ένα Media Pipeline, το οποίο αποτελείται από δύο διαφορετικά στοιχεία, τα WebRtcEndpoint και PlayerEndpoint. Θα δούμε το στοιχείο αυτό και στο επόμενο κομμάτι.

Στην ακόλουθη εικόνα φαίνεται το κουμπί Latest Rec, το οποίο, έχει επιλεγεί από τον χρήστη και πλέον μπορεί να αναπαράγει την τελευταία καταγεγραμμένη συνδιάσκεψη.



Εικόνα 35: Αναπαραγωγή τελευταίας καταγεγραμμένης συνδιάσκεψης

5.9. Λίστα προηγούμενων καταγραφών

Όλες οι καταγραφές ενός χρήστη εμφανίζονται σε μία λίστα, η μία κάτω από την άλλη, εφόσον κάποιος χρήστης πατήσει το κουμπί «Show past rec». Κάθε χρήστης μπορεί να δει μόνο τις δικές του καταγραφές και όχι των υπόλοιπων χρηστών. Επιλέγοντας μια καταγραφή από την λίστα ένας χρήστη μπορεί να την αναπαράγει.

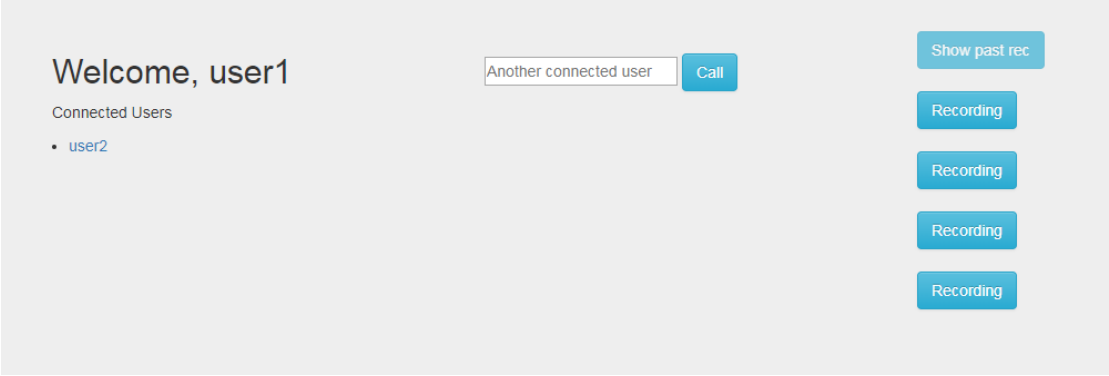
Η λίστα των προηγούμενων καταγραφών εμφανίζεται όταν ένα μήνυμα askRecordings φεύγει από τον client προς τον sever. Σε αυτή την περίπτωση καλείται η συνάρτηση askRecordings(), η οποία επιστρέφει μία λίστα με τα URL των προηγούμενων δέκα πιο πρόσφατων καταγραφών στις οποίες ο τρέχων χρήστης ήταν ένας από τους συμμετέχοντες. Για να επιστρέψει τις δέκα πιο πρόσφατες καταγραφές,

γίνεται ταξινόμηση όλων των καταγραφών του συγκεκριμένου με βάση το χρόνο καταγραφής τους.

Στη συνέχεια, φεύγει ένα μήνυμα showRecordings από τον server προς τον client και καλείται η αντίστοιχη συνάρτηση showRecordings() του client. Μέσω αυτής της συνάρτησης δημιουργείται ένα κουμπί με όνομα «Recording», το οποίο με το που πατηθεί δημιουργείται ένα αντικείμενο audio για την αναπαραγωγή του αρχείου καταγραφής.

Για να είναι εφικτή η αναπαραγωγή μίας καταγεγραμμένης συνδιάσκεψης, θα πρέπει να δημιουργηθεί ένα Media Pipeline, το οποίο σε αυτή την περίπτωση, θα πρέπει να αποτελείται από δύο διαφορετικά στοιχεία, τα WebRtcEndpoint και PlayerEndpoint. Η παράμετρος uri του Media Pipeline είναι το uri του εγγεγραμμένου αρχείου.

Στην ακόλουθη εικόνα φαίνεται ένα παράδειγμα, όπου εμφανίζεται μία λίστα με τις τέσσερα αρχεία καταγραφής για τον χρήστη user1.



Εικόνα 36: Λίστα με τις προηγούμενες καταγραφές για τον χρήστη user1.

6. Συμπεράσματα

Σκοπός της εργασίας αυτής ήταν ο σχεδιασμός και η υλοποίηση ενός απλού συστήματος που μπορεί να χρησιμοποιηθεί για την επικοινωνία σε πραγματικό χρόνο με την πραγματοποίηση κλήσεων βίντεο μεταξύ ιατρού - ασθενή. Η οπτικοακουστική επικοινωνία μεταξύ ιατρού – ασθενή, όπως είδαμε και παραπάνω, πραγματοποιείται μέσω μιας εφαρμογής η οποία ακολουθεί το μοντέλο αρχιτεκτονικής πελάτη-διακομιστή (client-server). Η εφαρμογή επικοινωνεί με δύο διακομιστές, τον Kurento Media Server και τη βάση δεδομένων MongoDB. Για να ελαχιστοποιήσουμε την καθυστέρηση στην επικοινωνία μεταξύ της εφαρμογής και των διακομιστών οι τελευταίοι μπορούν να βρίσκονται στο ίδιο φυσικό μηχάνημα με την εφαρμογή, όπως συμβαίνει στην περίπτωση μας ή να βρίσκονται σε διαφορετικά μηχανήματα στο ίδιο υποδίκτυο.

Η εφαρμογή που αναπτύχθηκε στην παρούσα εργασία είναι μια εφαρμογή single-page, δηλαδή είναι μια εφαρμογή ιστού που χωράει σε μια σελίδα με σκοπό να προσφέρει μια πιο ομαλή εμπειρία χρήσης σε σύγκριση με ένα πρόγραμμα στο υπολογιστή. Το σύστημα υλοποιήθηκε με επιτυχία και πλέον μπορεί να χρησιμοποιηθεί. Η υλοποίηση του συστήματος αυτού παρουσιάζει και εκμεταλλεύεται τις δυνατότητες της τεχνολογίας WebRTC. Πιο συγκεκριμένα, παρουσιάζει τον κύριο τρόπο χρήσης του WebRTC, που είναι η επικοινωνία σε πραγματικό χρόνο. Το WebRTC μπορεί επίσης να χρησιμοποιηθεί για τη μεταφορά δεδομένων μεταξύ συνδεδεμένων συνομηλίκων.

Κατά την χρησιμοποίηση του συστήματος αυτού δεν παρατηρήθηκε κάποιος από τους πιθανούς περιορισμούς που αναφέραμε στο κεφάλαιο 2. Ωστόσο, αυτοί είναι πιθανό να εμφανιστούν όταν ένας μεγαλύτερος αριθμός χρηστών λάβει μέρος στην επικοινωνία. Το συγκεκριμένο σύστημα δοκιμάστηκε με 2 - 3 χρήστες κάθε φορά, καθώς έχει σχεδιαστεί για την επικοινωνία γιατρού – ασθενή οπότε οι συμμετέχοντες δεν αναμένεται να είναι παραπάνω από δύο κάθε φορά.

Όσον αφορά το WebRTC και από την ανάλυση που έγινε στα προηγούμενα κεφάλαια, μπορούμε να συμπεράνουμε ότι το WebRTC έχει προάγει τον τομέα των επικοινωνιών σε σημαντικό επίπεδο. Εκτιμάται, ωστόσο ότι είναι μια πολλά

υποσχόμενη τεχνολογία στον χώρο των επικοινωνιών σε πραγματικό χρόνο και πρόκειται να αξιοποιηθεί σε μεγαλύτερο βαθμό στο μέλλον.

Αναμένεται ότι με την τεχνολογία WebRTC, οι χρήστες θα φθάσουν στο επίπεδο να επικοινωνούν μεταξύ τους χωρίς να χρειάζεται να κάνουν λήψη κάποιας εφαρμογής στον υπολογιστή, στο κινητό ή στο tablet τους δεδομένου ότι θα υπάρχει ένας σύγχρονος browser εγκατεστημένος στη συσκευή τους όποια και αν είναι αυτή.

Επιπλέον η τεχνολογία WebRTC προτιμάται και από τους προγραμματιστές διότι διευκολύνει το έργο τους. Δηλαδή απαιτείται λιγότερος χρόνος για να αναπτυχθεί μία εφαρμογή επικοινωνίας πραγματικού χρόνου.

Συνοψίζοντας, τα API και τα πρότυπα του WebRTC μπορούν να χρησιμοποιηθούν ως εργαλεία για τη δημιουργία εφαρμογών περιεχομένου επικοινωνίας. Τέτοιες εφαρμογές, μπορούν να χρησιμοποιούνται στη τηλεφωνία, την παραγωγή βίντεο, τη δημιουργία μουσικής, τη συλλογή ειδήσεων και πολλά άλλα.

7. Βιβλιογραφία

[1] Salvatore Loreto; Simon Pietro Romano, (2014). Real-Time Communication with WebRTC. Διαθέσιμο από τον διαδικτυακό τόπο: https://www.safaribooksonline.com/library/view/real-time-communication-with/9781449371869/ch01.html

[2] Pedro Rodríguez; Javier Cerviño; Irena Trajkovska; Joaquín Salvachúa, (2014). Advanced Videoconferencing Services Based on WebRTC. Διαθέσιμο από τον διαδικτυακό τόπο: https://www.researchgate.net/publication/235639869_Advanced_Videoconferencing_Services_Based_on_WebRTC

[3] Sam Dutton, (2012). Getting Started with WebRTC. Διαθέσιμο από τον διαδικτυακό τόπο: https://www.html5rocks.com/en/tutorials/webrtc/basics/#toc-mediastream

[4] Web browser. Διαθέσιμο από τον διαδικτυακό τόπο: https://en.wikipedia.org/wiki/Web_browser

[5] Using a Database (with Mongoose). Διαθέσιμο από τον διαδικτυακό τόπο: https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs/mongoose

[6] Mongoose. Διαθέσιμο από τον διαδικτυακό τόπο: http://mongoosejs.com/docs/guide.html

[7] Kurento Documentation, Release 6.6.2. Διαθέσιμο από τον διαδικτυακό τόπο: https://media.readthedocs.org/pdf/doc-kurento/stable/doc-kurento.pdf

[8] Recording WebRTC Sessions: client side or server side?. Διαθέσιμο από τον διαδικτυακό τόπο: https://bloggeek.me/recording-webrtc-sessions/

[9] Recording Video Sessions Using WebRTC. Διαθέσιμο από τον διαδικτυακό τόπο: http://www.oodlestechnologies.com/blogs/Recording-Video-Sessions-Using-WebRTC

[10] Kurento Tutorials. Διαθέσιμο από τον διαδικτυακό τόπο: http://doc-kurento.readthedocs.io/en/stable/tutorials.html

[11] Introducing Kurento. Διαθέσιμο από τον διαδικτυακό τόπο: http://doc-kurento.readthedocs.io/en/stable/introducing_kurento.html

[12] Kurento. Διαθέσιμο από τον διαδικτυακό τόπο: https://github.com/Kurento/doc-kurento-readthedocs/tree/master/source/tutorials

[13] WebRTC 1.0: Real-time Communication Between Browsers. Διαθέσιμο από τον διαδικτυακό τόπο: http://w3c.github.io/webrtc-pc/#media-stream-api-extensions-for-network-use

[14] HTML. Διαθέσιμο από τον διαδικτυακό τόπο: http://el.wikipedia.org/wiki/HTML

[15] JavaScript. Διαθέσιμο από τον διαδικτυακό τόπο: http://en.wikipedia.org/wiki/JavaScript

[16] CSS. Διαθέσιμο από τον διαδικτυακό τόπο: http://en.wikipedia.org/wiki/Cascading_Style_Sheets

[17] What is MongoDB?. Διαθέσιμο από τον διαδικτυακό τόπο: https://www.mongodb.com/what-is-mongodb

[18] MongoDB. Διαθέσιμο από τον διαδικτυακό τόπο: https://en.wikipedia.org/wiki/MongoDB

[19] Network address translation. Διαθέσιμο από τον διαδικτυακό τόπο: https://en.wikipedia.org/wiki/Network_address_translation

[20] Interactive Connectivity Establishment. Διαθέσιμο από τον διαδικτυακό τόπο: https://en.wikipedia.org/wiki/Interactive_Connectivity_Establishment

[21] STUN. Διαθέσιμο από τον διαδικτυακό τόπο: https://en.wikipedia.org/wiki/STUN

[22] Traversal Using Relays around NAT. Διαθέσιμο από τον διαδικτυακό τόπο: https://en.wikipedia.org/wiki/Traversal_Using_Relays_around_NAT